גרפיקה מתקדמת

עבוד נתונים

Commodore 64

ס.גריי מ.הולמס

ארכימדס

BUG

commodore
64

גרפיקה מתקדמת

ס.גריי מ.הולמס

ארכימדס – הוצאה לאור

ספר זה תורגם מ:

WATSON'S WORKBOOK

EXPLORING GRAPHICS

by

S. GRAY & M.D. HOLMES

ISBN 0 907792 50 2

תרגום: שי זמיר

תוכן העניינים

הקדמה

כל ספר מסדרת ספרי העבודה של ווטסון עוצב כדי לתאר צד יחודי של תכנות מחשבים. בהמשך למסורת זו, "גרפיקה מתקדמת" נכתב במיוחד כדי ללמד את הקורא את יסודות הפונקציות הגרפיות של הקומודור 64. הוא מציע לא רק הכרות עם שימוש בגרפיקה עבור המתכנת המתחיל, אלא גם למתכנת בעל הידע. שתי תוכניות שרות (UTILITES) מצורפות כדי לעזור למתכנת לעצב תוים וספרייטים משלו. כמו-כן מצורף מדריך צעד אחרי צעד לשימוש בגרפיקה בקצב שכל הגילים יוכלו לעקוב אחריו בקלות.

כדי לתאר נקודות מסוימות, מפותחות ארבע תוכנות עקריות בספר זה. ואלו הן: BREAROUT (משחק), CHAR.GEN (תוכנית שרות), SPRITE.GEN (תוכנית שרות נוספת) ו-TARGET (משחק ספרייטים). התוכניות נכתבו בשלבים כדי להציג בהדרגה פקודות חדשות והשמוש בהן. התוכניות הקטנות משתלבות בתוכניות הסבר אחרות, כדי ליצור את התוכנית הסופית והמפורטת אשר מציגה את האפשרויות הגרפיות של הקומודור 64.

תוכניות אלו יספקו תמריץ לקוראים המתקדמים יותר ואשר בעזרת ספר זה יוכלו לפתח תוכניות גרפיות משלהם.

התוכנית הראשונה שנכתוב היא גרסה של המשחק הנודע "BREAKOUT" אשר בו המטרה היא להוריד לבנים מן הקיר בעזרת כדור, אשר פוגע במחבט הנשלח על-ידי המשתמש. תוכנית זו מפותחת בשלבים כדי להראות באיזו קלות אפשר לפתח משחק מסובך.

תוכנית עצוב התוים מספקת אמצעים מלאים הדרושים למשתמשים כדי לעצב קבוצת תוים משלהם וכן עושה את החיים קלים יותר לאלו שאינם מרוצים מקבוצת התוים הסטאנדרטית של הקומודור 64.

תוכנית עצוב הספרייטים מאפשרת למשתמשים לעצב בקלות ספרייטים משלהם. ההיקף המלא של האפשרויות של תוכנית זו נחשף במשחק הספרייטים הנכלל גם הוא בספר זה.

קבוצת התוים המיוחדת של ד"ר ווטסון

בעלי קומודור 64 אשר ברשותם מדפסת יודעים עד כמה קשה להדפיס את תוי השליטה המיוחדים של הקומודור. במאמץ להתמודד עם קושי זה אבחר קבוצת תוים מיוחדת עבור ספר הלימוד של ווטסון על הקומודור 64. קבוצה זו תתאר את תו השליטה עם קיצור המילים אשר בעזרתן נוצר התו. למשל, תו ה"סמן מעלה" מתואר על-ידי <UCRSR>, הסוגריים המשולשים משמשים כמפרידים עבור כל תאורי התוים המיוחדים.

תאורים מיוחדים אלו ישמשו בכל הרשומות (LISTINGS) כאשר תוים מיוחדים נדרשים. כאשר אתה מכניס את הרשומות, אל תכניס את תאור התו המיוחד כלשונו, ציית לתאור. כלומר במקרה של <UCRSR> לחץ על מקש ה-SHIFT ועל מקש UP-CURSOR (=סמן מעלה).

נספח אחד מכיל טבלה המראה את כל התוים המיוחדים. במקרה של ספק עיין בטבלה.

פרק 1
גרפיקה של תווים

הקומודור 64 הוא יחיד במינו מבין שאר המחשבים הביתיים. אחת הסיבות העיקריות לכך היא העדרן הנראה של פקודות גרפיות. למחשב הדראגון 15 פקודות, לספקטרום - ארבע-עשרה. הקומודור 64 בא מצויד בשנים בלבד. אך שתי פקודות אלו הן חזקות כל כך שהאפשרויות הגרפיות של הקומודור 64 עולות על אלו של רבים מן המחשבים הביתיים האחרים. שתי הפקודות המהוללות הן: POKE ו-PEEK.

POKE

בעיקרו של דבר, פקודת ה-POKE ממקמת מידע היישר לתוך כתובת הזכרון. הפקודה לשים (POKE) מספר בזכרון חייבת לציין את המספר יחד עם כתובת הזכרון. לדוגמא, הפקודה:

POKE 828,90

תשים את המספר 90 לתוך כתובת הזכרון 828. כתובת הזכרון 828 היא חלק מ"מחסנית" הקסטה (CASSETE BUFFER) והוא שימושי רק בזמן שטייפ הקסטות בפעולה.

פונקציות רבות של הקומודור 64 נשלטות על-ידי תוכנן של כתובות זכרון מסוימות. למשל, הצבעים שאתה רואה על המסך נקבעים על-ידי תוכנן של שלוש כתובות זכרון, 53280, 53281 ו-646. כדי לחקור זאת, אפשר לקבוע לולאה. לולאה זו תרוץ דרך כל הצבעים האפשריים. לקומודור 64 יש שישה-עשר צבעים (0 עבור שחור עד 15 עבור אפור) כך שהלולאה תנוע בין 0 ל-15.

תוכנית 1.1(a)

```
10 FOR X=0 TO 15
20 POKE 53281,X
30 FOR D=1 TO 500:NEXT D
40 NEXT X
```

כשתוכנית זו תרוץ, מסך הטלויזיה שלך יהבהב דרך כל טור הצבעים.

53281 הוא המיקום בזכרון השולט על צבע רקע המסך. צבע הגבול (BORDER) נשלט על-ידי 53280. כדי להדגים זאת, הוסף שורה 20 חדשה והרץ שנית את התוכנית.

תוכנית (1.1(b

```
20 POKE 53280,X
```

ועכשיו כמו שציפינו, הגבול יהבהב בשישה-עשר הצבעים. הכתובת השלישית אשר עוסקת בסמן היא 646. כדי לראות מה כתובת זו עושה, שנה את שורה 20, הוסף את שורה 25 והרץ את התוכנית.

תוכנית (1.1(c

```
20 POKE 646,X
25 PRINT"THIS IS COLOUR";X
```

כעת תראה את הסמן משנה את צבעו. כאשר התוכנית תסתיים לא תוכל לראות את הודעת "READY" מכיוון שצבע ה"דיו" x הוא אפור וכך גם צבע הרקע. אם צבע הרקע וצבע הדיו הם שוים הרי שהסמן אינו נראה.

כדי להפוך את הסמן לנראה, הכנס את הפקודה הישירה הבאה:

POKE 646,2

פקודה זו תהפוך את הסמן לאדום וכתוצאה מכך ניתן לראיה.
הזהר כשאתה מכניס את הפקודות משום שאינך יכול לראות את מה שאתה מדפיס וקל מאוד לעשות טעות. הזהר מאוד בכל תו ובמיוחד במספרים. אם אחרי לחיצת RETURN הסמן לא הופיע הרי שעשית טעות ואתה חייב להכניס את השורה מחדש.

עכשיו כשאנחנו מכירים את כתובות הרקע, הגבול והדיו (53281, 53280 ו-646 בהתאמה) אנו יכולים לכתוב תוכנית אשר תקבע את צבעי המסך בכל צורה שנרצה. התוכנית הבאה מבצעת בדיוק את הדבר הזה.

תוכנית 1.2

```
10 INPUT"FOREGROUND COLOUR";FC
20 IF FC<0 OR FC>15 THEN PRINT"<UCRSR>":GOTO 10
30 INPUT"BACKGROUND COLOUR";BC
40 IF BC<0 OR BC>15 THEN PRINT"<UCRSR>":GOTO 30
50 INPUT"INK COLOUR";IC
60 IF IC<0 OR IC>15 OR IC=FC THEN PRINT"<UCRSR>":
GOTO 50
70 POKE 53281,FC:REM FOREGROUND
80 POKE 53280,BC:REM BORDER
90 POKE 646,IC:REM INK
```

PEEK

הפקודה-האחות לפקודת POKE היא פקודת PEEK. בעוד POKE מאפשר לנו לשים מידע בזכרון, PEEK מאפשר לנו להחזיר מידע. למעשה פקודת PEEK מתבוננת לתוך הכתובת המצויינת ומעתיקה את מה שהיא מוצאת שם. לדוגמא:

PRINT PEEK(828)

פקודה זו תציג את תוכנה של כתובת 828. אלא אם כן כיבית את המחשב או העלית תוכנית מאז ההסבר על POKE, ערכה של כתובת הזכרון 828 יהיה 90.

כתובת הזכרון בפקודה (PEEK) חייבת להיות בתוך סוגריים.

ישנה עוד תכונה מוזרה של PEEK שחשוב לדעת. הכנס את הפקודות הישירות הבאות:

POKE 53280,4:PRINT PEEK(53280)

המספר 244 הוא לא בדיוק מה שציפינו. כדי לקבל את התוצאה הרצויה של 4 אנו זקוקים לפעולת "וגם" (AND) בין 244 ו-15 (חמישה עשר הוא ערך הצבע הגבוה ביותר של x). למעשה נערכת פעולת "וגם" בין הערך הבינארי של 244 לערך הבינארי של 15.

$$\begin{array}{r} 1\,1\,1\,1\,0\,1\,0\,0 \\ \underline{0\,0\,0\,0\,1\,1\,1\,1_2} \\ 0\,0\,0\,0\,0\,1\,0\,0_2 \end{array}$$

תרשים המראה פעולת "וגם" (AND) בין 244 ו-15

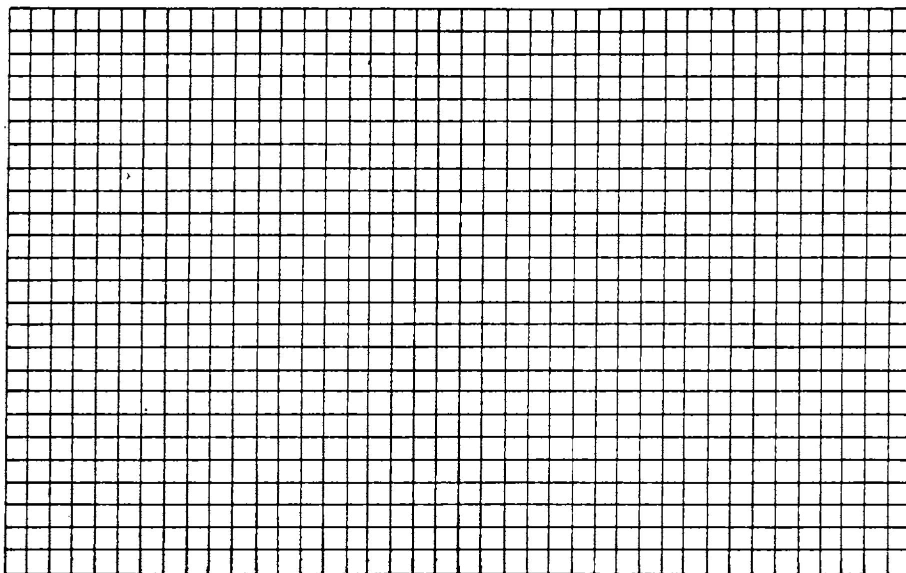
פעולת 'AND' פועלת על-ידי נטילת כל סיבית של 244 והשוואתה לסיבית התואמת של 15. אם שניהן 1 הרי שהתוצאה היא חיובית (1). אם לא - הרי שהתוצאה היא שלילית (0).

ראינו כיצד כתובת הזכרון 53280 מאחסנת את הצבע סגול לא כ-4 אלא כ-244. ביכולתנו לאחסן (POKE) את הערך 244 לכתובת 53280 כדי לקבל צבע גבול סגול.

עד כה ראינו כיצד לשנות שלוש כתובות צבע של הקומודור 64. אפשר לשנות את הצבע של כל תא במסך בנפרד במקום שינוי צבע המסך כולו.

אחת מתכונותיו של מסך הקומודור 64 היא שהוא ממופה מסך. משמעות הדבר שלכל תו או תא במסך יש כתובת בזכרון הקשורה לו. הקומודור 64 שונה בזה שלכל תא במסך יש שתי כתובות בזכרון הקשורות אליו. אחת שולטת בתו אותו יכיל התא ואחת מכתיבה את צבעו של התא.

מסך הקומודור 64 מחולק ל-1000 תאים - 40 לרוחב ו-25 למטה.



המסך ממופה הזכרון של הקומודור 64

הפינה השמאלית העליונה (הידועה בתור נקודת הבית HOME) היא כתובת הזכרון 1024. התא הבא בשורה הוא 1025 וכן הלאה. מיקום הזכרון עבור התחתית הימנית היא 2023.

כתובות בצבע בזכרון מתחילות ב-55296 עבור הפינה השמאלית עליונה ומסתיימות ב-56295 עבור הפינה התחתונה מימין. כדי להדגים זאת הכנס את התוכנית הבאה:

תוכנית 1.3(a)

```
10 POKE 53281,1:REM SCREEN COLOUR WHITE
20 POKE 1024,1:REM CHARACTER A
30 POKE 55296,2:REM COLOUR RED
```

בשורה 20 "A" הוכנס לזכרון המסך והאות "A" הופיעה על המסך. קודי מסך אלו עבור POKE הם גרסת קומודור לקוד ASCII וכל הקודים הללו מצויים ברשימה בנספח שניים. כל התוים יכולים להחקר על-ידי שנויי התוכנית כך שהיא תעבור דרך כולם.

תוכנית 1.3(b)

```
10 POKE 53281,1
15 FOR X=0 TO 255
20 POKE 1024+X,X
30 POKE 55296+X,2
40 NEXT X
50 PRINT"<HOME><10DCRSR>"
```

שמוש בלולאת FOR-NEXT הוא פשוט ונוח לאחסון (POKE) ערכים לכל תא במסך

תוכנית 1.3(c)

```
10 FOR X=0 TO 999
20 POKE 1024+X,83
30 POKE 55296+X,2
40 NEXT X
```

תוכנית זו תמלא כל תא במסך בלב אדום.

בפרק זה נפתח משחק ווידאו פשוט המבוסס על המשחק "BREAKOUT", בו המשחק צריך להרוס חומה על-ידי הקפצת כדור ממחבט. המשחק יפותח בשלבים והדבר תלוי בך אם לעבור את השלבים או להעתיק את התוכנית כולה בסוף הפרק. בכל אופן, מומלץ שהקוראים יעיינו בשלבים השונים של כתיבת המשחק.

כדור נע

ודאי שהחלק החשוב ביותר יהיה חקוי הכדור הנע. גם מסע ארוך מתחיל בצעד קטן, לכן נתחיל על-ידי הנעת הכדור לרוחב חלקו העליון של המסך. אם נבחר בכדור כמו התו "O" (POKE 81), נוכל להניעו לרוחב המסך על-ידי פקודת POKE:

תוכנית 1.4

```
10 PRINT"<CLR>":POKE 53281,1
20 FOR X=0 TO 39
30 POKE 1024+X,81
40 POKE 55296+X,2
50 NEXT X
```

כאשר התוכנית מורצת היא מציירת שורה של כדורים אדומים לרוחב המסך ומשאירה זנב מאחוריה. כעת אנו צריכים למחוק זנב זה. דבר זה ניתן לעשות בקלות על-ידי אחסון רווח (ערך POKE של 32) מאחורי הכדור, ועל-ידי כך נודא כי רק כדור אחד יהיה על המסך באותו הזמן. אחסון נוסף זה יכול להכלל בלולאה אך צריך להזהר ולהבטיח כי מיקומו יהיה תמיד מאחורי הכדור. כלומר, הכדור ב-1024+x והרווח ב-1024+x-1

תוכנית 1.4(a)

```
10 PRINT"<CLR>":POKE 53281,1
20 FOR X=0 TO 39
30 POKE 1024+X,81
40 POKE 55296+X,2
45 POKE 1024+X-1,32
50 NEXT X
```

תוכנית 1.4(b) היא תוכנית דומה, אך כעת הכדור נע באלכסון מפסגתו השמאלית של המסך לכיוון התחתית הימנית.

תוכנית 1.4(b)

```
10 PRINT"<CLR>":POKE 53281,1
20 FOR X=0 TO 24
30 POKE 1024+X+X*40,81
40 POKE 55296+X+X*40,2
50 POKE 1024+X-1+(X-1)*40,32
60 NEXT X
```

מכיון שקיימים ארבעים מקומות לרוחב (0-39), הקואורדינטות יכולות להשתנות על-ידי הוספת ארבעים בכל פעם שאנו רוצים לנוע מטה. כדי לנוע מעלה ברור שנחסיר ארבעים.

הכדור נעצר בשורה התחתונה כדי למנוע אחסון דבר מה בזכרון שאינו זכרון מסך, זהירות המשתלמת תמיד.

עד כה כל תנועות הכדור נשלטו על-ידי משתנה לולאה. דבר זה מגביל אותנו בכך שהדבר מאפשר לכדור לנוע בכיוון העכשווי בלבד. כדי לעקוף זאת, שני משתנים ישמשו לשליטה בקואורדינטות x ו-y. שני משתנים אלה קרויים x ו-y בהתאמה. הבעיה של מחיקת הזנב שהכדור משאיר נפתרת על-ידי הצגת שני המשתנים xx ו-yy אשר זוכרים את מיקום הכדור לפני שהוא זז הלאה.

תוכנית 1.4(c)

```
10 PRINT"<CLR>":POKE 53281,1
20 X=0:Y=0
30 POKE 1024+X+Y*40,81
40 POKE 55296+X+Y*40,2
50 YY=Y:XX=X
60 Y=Y+1:X=X+1
70 IF Y=25 THEN STOP
80 POKE 1024+XX+YY*40,32
90 GOTO 30
```

שורה 70 מפסיקה את התוכנית לפני ש-y עובר את גבולותיו. עד כה יצרנו כדור אשר נע בכיוון אחד עד אשר יגיע לערך קיצוני ואז תעצר התוכנית. מה שדרוש הוא כדור אשר ינתר כאשר x ו/או y נמצאים בערכים הקיצוניים שלהם. אפקט זה יכול להווצר על-ידי שימוש בשני משתנים נוספים x1 ו-y1.

כאשר כדורנו נע משמאל לימין, ערכו של x צריך לגדול כך ש-x1 יהיה 1+. אך כשאנו רוצים שהכדור ינוע מימין לשמאל הרי ש-x צריך לקטון וכך x1 יהיה שווה ל 1-. ואותם חוקים יתפסו עבור y ו-y1. תוכנית 1.5 משתמשת במשתנים x1 ו-y1. שורות 70 ו-80 בודקות את x ו-y עבור ערכים קיצוניים, כאשר אחד מהם בערכו הקיצוני. מתוקן ערך ההעלאה המתאים.

תוכנית 1.5

```
10 PRINT"<CLR>":POKE 53281,1
20 X=0:Y=0:X1=1:Y1=1
30 POKE 1024+X+Y*40,81
40 POKE 55296+X+Y*40,2
50 YY=Y:XX=X
60 Y=Y+Y1:X=X+X1
70 IF X>38 OR X<1 THEN X1=-X1
80 IF Y>23 OR Y<1 THEN Y1=-Y1
90 POKE 1024+XX+YY*40,32
100 GOTO 30
```

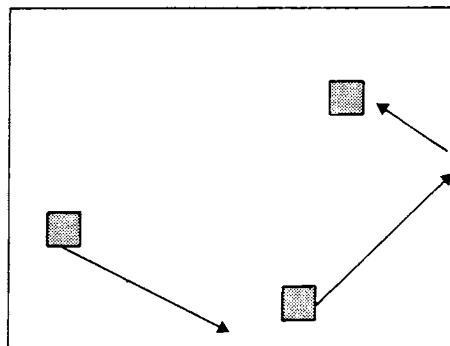
הערכים הראשוניים של x1 ו-y1 הם 1+ כך שהכיוון הראשוני הוא תנועה אלכסונית מהפסגה השמאלית אל התחתית הימנית. כאשר הכדור מנתר הוא תמיד מנתר באלכסון. הערכים ההתחלתיים של x1 ו-y1 לא רק נותנים לכדור כיוון התחלתי אלא גם שולטים בצורה שהכדור יקפוץ. כדי להדגים זאת נשנה את שורה 20 כך שהיא תראה:

```
20 x=0:y=0:x1=1:Y1=0
```

כאשר תריץ שוב את התוכנית הכדור פשוט ינוע לרוחב פסגת המסך. הערך y ו-y1 לעולם לא משתנים משום ש-y+0=Y כאשר y1=0. כך שעד כה תוכניתנו תקפיץ כדור באלכסון. השיפור יהיה קפיצה מקרית אשר תאפשר לנוע באלכסון ולמעלה או למטה או ימינה ושמאלה.

כדור הקופץ אקראית

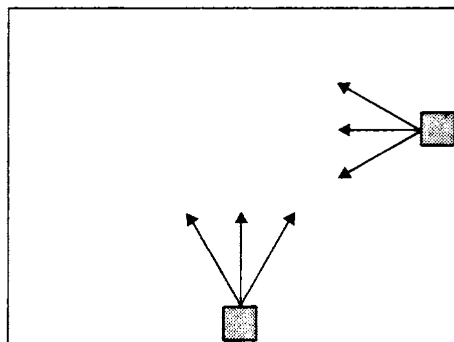
ברגע זה כדורנו הנע אלכסונית מנתר באופן המתואר בציור אחד.



ציור אחד

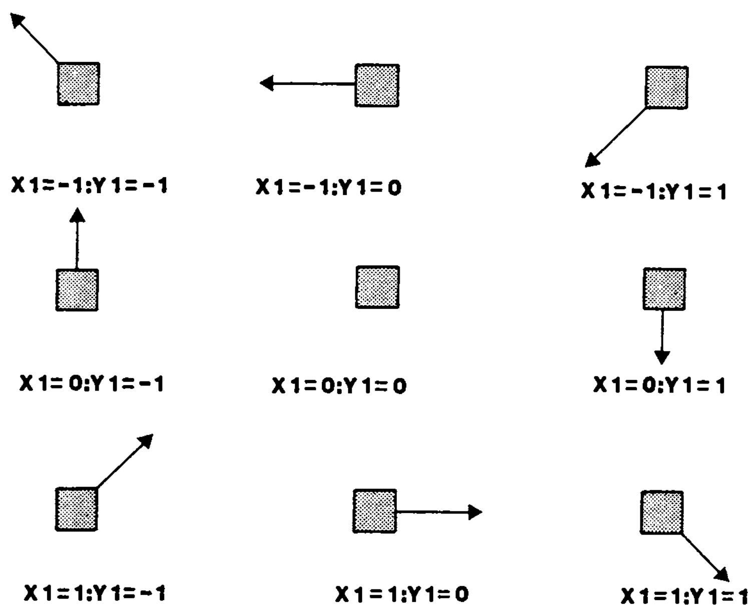
כדור הקופץ אלכסונית

תאר לעצמך כמה ישתפר הכדור הנע שלנו אם בכל פעם שהוא פוגע בקצה הוא יוכל לנתר באחד משלושה כיוונים, כפי שמתואר בציור שנים.



ציור שניים

כדי להפיק סוג זה של קפיצה, ערכי x1 ו-y1 צריכים להשתנות באופן אקראי בין הערכים 1-, 0 ו +1.



ציור שלוש

ציור שלוש מדגים את 9 כיווני התנועות האפשריות של הכדור. למעשה ישנם רק שמונה כיוונים. כאשר x1 ו-y1 שוים שניהם לאפס הכדור אינו נע. חייבים להקפיד ולהזהר שדבר זה לא יקרה.

השנויים הדרושים בערכי x1 ו-y1 יעשו בשתי שגרות. שגרה אחת עבור שנוי כיוון x ואחת עבור y.

כדי לקבל מספר אקראי בטווח 1- עד 1 אנו זקוקים ראשית לחשב מספר אקראי בטווח שבין 1 ל-3, ואז אם הערך האקראי הוא שלוש נשנה אותו ל-1- ואם הוא 2 נשנה אותו ל-0. דבר זה נעשה בשורות 1020-1050 ו-2020-2050.

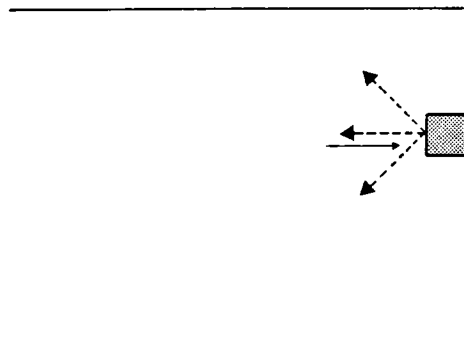
```
20 X=10:Y=10:X1=1:Y1=1
70 IF X>38 OR X<1 THEN GOSUB 1000
80 IF Y>23 OR Y<2 THEN GOSUB 2000

1000 REM CHANGE Y DIRECTION
1010 X1=-X1:REM X DIRECTION
1020 Y1=INT(RND(1)*3)+1
1030 IF Y1=3 THEN Y1=-1:RETURN
1040 IF Y1=2 THEN Y1=0:RETURN
1050 Y1=1
1060 RETURN

2000 REM CHANGE X DIRECTION
2010 Y1=-Y1:REM Y DIRECTION
2020 X1=INT(RND(1)*3)+1
2030 IF X1=3 THEN X1=-1:RETURN
2040 IF X1=2 THEN X1=0:RETURN
2050 X1=1
2060 RETURN
```

חשוב להבין את ההגיון מאחורי הקפיצה כאן.

נניח כי הבאנו את הכדור לערכו הימני הקיצוני של x כפי שמתואר בציור ארבע.



ציור ארבע

ערכו של x1 צריך להשתנות כך שהכדור ינוע אחורה ושמאלה. ערכו של y נבחר בצורה אקראית כדי לראות אם הכדור ינוע באלכסון אל התחתית השמאלית של המסך (x1=-1:y1=+1), בקו ישר אחורה לרוחב המסך (x1=-1:y1=0) או באלכסון בכיוון פסגת המסך (x1=-1:y1=-1). אותם חוקים תופסים כאשר ציר y הגיע לקצה המסך.
אם תריץ את תוכנית 1.6 ותשאיר אותה לכמה זמן, לבסוף הכדור ירוץ אל אחת הפינות, ובנקודה זו x ו-y בערכים הקיצוניים שלהם, כך ששתי השגרות נקראות זו אחר זו. דבר זה עלול לגרום לתוכנית ליפול. כאשר אתה מבצע POKE לכל ערך אשר יוצא מגבול הפעילות הפנימית של הקומודור הנפילה יכולה להיות בכל צורה ואתה עלול למצוא שאינך יכול להכניס דבר מן המקלדת. כאשר הקומודור 64 נופל הדבר הטוב ביותר הוא לכבות ולהדליק שוב וכך נקבעות מחדש כל כתובות הזכרון. בשגרה הראשונה, כיוון x משתנה ו-y חדש נבחר אקראית. כעת השגרה השניה נקראת אשר קובעת מחדש את כיוון y ומשנה שוב את כיוון x וכך, לדוגמא אם הכדור נמצא בפינה השמאלית העליונה, השגרה הראשונה תשנה את x1 ל-1. ואז השגרה השניה יכולה לשנות אקראית את x1 חזרה ל-1-, דבר שיגרום לכדור לנוע מחוץ למסך וכך לגרום לנפילת התוכנית.
בעיה זו יכולה להפתר בקלות על-ידי שילוב שתי השגרות לאחת בלבד. שורה 80 צריכה להמחק ושורה 70 להשתנות ל-

```
70 IF X>38 OR X<1 OR Y>23 OR Y<2
THEN GOSUB 1000
```

כעת השגרות ב-1000 ו-2000 צריכות להמחק ושגרה חדשה להכנס.

תוכנית 1.6(b)

```
1000 REM CHANGE DIRECTION
1010 IF X>38 OR X<1 THEN X1=-X1
1020 IF Y>23 OR Y<2 THEN Y1=-Y1
1030 T=INT(RND(1)*3)+1
1040 IF T=1 THEN Y1=-1
1050 IF T=2 THEN Y1=0
1060 IF T=3 THEN Y1=1
1070 IF Y+Y1>23 THEN Y1=-1
1080 IF Y+Y1<2 THEN Y1=1
```

```
1090 T=INT(RND(1)*3)+1
1100 IF T=1 THEN X1=-1
1110 IF T=2 THEN X1=0
1120 IF T=3 THEN X1=1
1130 IF X+X1>38 THEN X1=-1
1140 IF X+X1<1 THEN X1=1
1150 RETURN
```

שגרה ארוכה למדי זאת מחליפה את שתי הישנות יותר. שורות 1030-1060 ו-1090-1120 צריכות להיות מוכרות לך, הן בוחרות את הכיוון באותה הדרך כמו קודם. החידוש האמיתי בשגרה זו הן הבדיקות בשורות 1070, 1080, 1130 ו-1140. אלו מבטיחות שהערכים החדשים של X1 ו-Y1 לא יגרמו לכדור לנוע מחוץ לגבולות המסך.

תוכנית הכדור הקופץ אקראית

```
10 PRINT"<CLR>":POKE 53281,1
20 X=10:Y=10:X1=1:Y1=1
30 POKE 1024+X+Y*40,81
40 POKE 55296+X+Y*40,2
50 YY=Y:XX=X
60 Y=Y+Y1:X=X+X1
70 IF X>38 OR X<1 OR Y>23 OR Y<2
THEN GOSUB 1000
90 POKE 1024+XX+YY*40,32
100 GOTO 30

1000 REM CHANGE DIRECTION
1010 IF X>38 OR X<1 THEN X1=-X1
1020 IF Y>23 OR Y<2 THEN Y1=-Y1
1030 T=INT(RND(1)*3)+1
1040 IF T=1 THEN Y1=-1
1050 IF T=2 THEN Y1=0
1060 IF T=3 THEN Y1=1
1070 IF Y+Y1>23 THEN Y1=-1
1080 IF Y+Y1<2 THEN Y1=1
1090 T=INT(RND(1)*3)+1
1100 IF T=1 THEN X1=-1
1110 IF T=2 THEN X1=0
1120 IF T=3 THEN X1=1
1130 IF X+X1>38 THEN X1=-1
1140 IF X+X1<1 THEN X1=1
1150 RETURN
```

התוכנית שעד כה כתבנו מקפיצה את הכדור מכל קצה של המסך. אך עבור המשחק הסופי הכדור חייב לקפוץ מן התחתית רק אם המחבט נמצא במקום. כמו-כן כאשר הכדור מגיע לחומה, הקפיצה תלויה בכמות הלבנים שנהרסו.

מה שדרוש היא פונקציה אשר תאפשר לנו להתבונן בכתובת זכרון מסך מסוימת (במיוחד זו אשר אליה עומד הכדור לנוע) ולדווח חזרה את אשר ראינו: פקודת PEEK!

תוכנית 1.7 מדגימה כיצד להשתמש ב-PEEK כדי להקפיץ כדור משורה המצויירת על קוד התו 160 (משבצת מלאה).

כאשר תוכנית זו מורצת. היא עשויה להראות דומה מאוד לזו הקודמת בפרק זה, אך הקפיצה נגרמת כתוצאה מהגיון אחר. קודם לכן, ערכנו בדיקה בלתי פוסקת של ערכי 'x' ו-'y' וכאשר הם הגיעו לערכים הקיצוניים שלהם התערבנו בקפיצה. לעומת זאת, בתוכנית 1.7 שורה 115 בודקת את הכתובת אליה אנו עומדים לנוע כדי לראות אם יש משהו בדרכנו. אם אין כלום, נניע את הכדור כמו בעבר, אך אם יש משהו (במקרה זה A=0) הרי איננו יכולים לאפשר לכדור לנוע לנקודה זו. שורה 116 משנה את משתנה הגדלת השורה Y1 והתוכנית מדלגת אחורה לשורה 60 כדי למצוא על מיקום חדש.

תוכנית 1.7

```
10 REM ENDLESSLY BOUNCING BALL
15 POKE 53281,1
20 PRINT "<CLR>"
21 FOR X=0 TO 39
22 POKE 1144+X,160:POKE 55416+X,0
23 NEXT X
30 X=10:Y=10
40 X1=1:Y1=1
50 POKE 1024+X+Y*40,81
60 POKE 55296+X+Y*40,2
70 XX=X:YY=Y
80 IF X>38 OR X<1 THEN X1=-X1
90 IF Y>23 OR Y<1 THEN Y1=-Y1
100 X=XX+X1
110 Y=YY+Y1
115 A=PEEK(55296+X+Y*40) AND 15
116 IF A=0 THEN Y1=-Y1:GOTO 100
120 POKE 1024+XX+YY*40,32
130 GOTO 50
```

אם לא היינו משתמשים ב-"AND 15" בשורה 115, ערכו של 'A' היה 224. החלק של AND 15 נותן את הערך המספרי הנמוך של הצבע בכתובת זו.

ההגיון מאחורי הקפיצה נשלט על-ידי מבט אל צבעו של הבית אליו אנו עומדים לנוע. אם הצבע הינו 0 (צבע החומה שלנו) אנו נעים. לעומת זאת אם נאחסן רווחים במקום משבצות מלאות (תוכנית 1.7(a)) ובכך נהפוך את השורה לבלתי נראית, הרי הכדור ימשיך לקפוץ כאילו החומה עדיין שם.

תוכנית 1.7(a)

```
22 POKE 1144+X,32:POKE 55416+X,0
```

הסיבה לכך היא שכתובות הצבע שלנו עדיין שחורות, כך שדרוש ערך שונה של PEEK, זה של התו שלפנים. אם הבית לפנים הוא חלק מן החומה, הרי ערך ה-PEEK יהיה 160 - קוד התו עבור משבצת מלאה. כך שעבור "משחק ההקפצה המושלם" אנו צריכים לשנות את שורה 22 בחזרה ולהחליף את שורה 115 ו-116 עם:

תוכנית 1.7(b)

```
22 POKE 1144+X,160:POKE 55416+X,0

115 A=PEEK(1024+X+Y*40)
116 IF A=160 THEN Y1=Y1*-1:GOTO 100
```

כעת בידינו שתי דרכים לגרום לכדור לקפוץ. ראשית על-ידי מעקב רצוף על ערכי x ו-y ושנית באמצעות פקודת PEEK.

חמושים בידע זה ובנסיון (בלי להזכיר את פקודות PEEK ו-POKE) נוכל להתחיל ברצינות לפתח משחק. לפתיחה מחודשת, אחרי שמירת התוכנית אם ברצונך בכך, הקש NEW ונתחיל ברצינות.

המחבט ינוע מצד אל צד לאורך השורה התחתונה של המסך. מקש '1' משמש להנעת המחבט שמאלה, ומקש '9' - להנעתו ימינה. ברור שאי-אפשר להשתמש בפקודת INPUT מכיוון שהתוכנית תעצור אחרי כל קלט ותחכה עד שילחץ מקש RETURN. במקומה, משתמשים בפקודת GET אשר משמעותה היא שתוים יכולים להכנס ישירות מן המקלדת בלי צורך בלחיצה על RETURN. כדי להדגים זאת הכנס את התוכנית שלמטה:

תוכנית 1.8

```
10 PRINT"<CLR>"
20 GET A$:IF A$="" THEN 20
30 IF A$="1" THEN PRINT "LEFT"
40 IF A$="9" THEN PRINT "RIGHT"
50 GOTO 20
```

התוכנית לא תעזוב את שורה 20 אלא אם כן נלחץ על מקש כלשהו. מתעלמים ממקש זה אם איננו '1' או '9'.

עבור המשחק האמור, המחבט יהיה באורך שני תוים והוא ינוע לאורך השורה התחתונה של המסך (כתובות 1984-2023) תחת שליטתם של המקשים '1' ו-'9'. המשתנה 'BC' ישמש כמשתנה הטור של המחבט ו-'CB' יזכור את המיקום בו שהה המחבט (בדומה ל-'x' ו-'xx' בתוכנית הנעת הכדור). שיפור נוסף מתוסף כדי שהתוכנית תוודא שהמחבט נשאר על המסך. כעת אנו בודקים ש-BC ינוע בין 0 ל-38 כולל (ולא 39 משום שאורך המחבט שני תוים).

תוכנית 1.8(a)

```
10 REM MOVING BAT
20 PRINT "<CLR>":POKE 53281,1
30 GET A$:IF A$="" THEN 30
40 IF A$="9"AND BC<38 THEN BC=BC+2:GOTO 70
50 IF A$="1" AND BC>0 THEN BC=BC-2:GOTO 70
60 GOTO 30
70 POKE 1984+CB,32 : POKE 1984+CB+1,32
80 POKE 1984+BC,160: POKE 1984+BC+1,160
90 POKE 56256+BC,0 : POKE 56256+BC+1,0
100 CB=BC
110 GOTO 30
```

בניית החומה

תוך שימוש בהגיון שהודגם בתוכנית 1.7(a) נבנה חומה רבת צבעים. שתי השורות העליונות של המסך ישמשו להצגת התוצאה העדכנית ומידע דומה נוסף.

תכנית 1.9

```
5 REM DRAW THE WALL
10 PRINT"<CLR>":POKE 53281,1
20 FOR X=0 TO 39
30 FOR Y=2 TO 7
40 POKE 1024+X+Y*40,160
50 POKE 55296+X+Y*40,Y
60 NEXT Y
70 NEXT X
80 PRINT"<9DCRSR>"
```

הריסת החומה

כעת, כשהחומה נבנתה, צריך לחשוב כיצד הכדור יהרוס אותה! הלבנים תוסרנה על-ידי אחסון תו הרווח בכתובת הנכונה. כדי לנסות זאת הכנס את הפקודה הבאה:

POKE 1164,32

ולבנה תעלם מן המסך. כדי להפוך את המשחק למעט קל יותר, הלבנים יוסרו שתים בבת-אחת. צריך, לעומת זאת, להבטיח שמשפטי ה-POKE יהיו רק עבור טורים 2,4,6 וכן הלאה עד 38. כלומר, משתנה הטור יהיה מספר זוגי!

נניח כי החומה עדיין שלמה והכדור מתקרב אל הטור העשירי. מכיון שעשר הוא מספר זוגי, אין שום בעיה. אנו פשוט מסירים חלק זה של החומה ומקפיצים את הכדור חזרה. לעומת זאת, אם הכדור מתקרב לטור השלושה-עשר הדבר אינו כה פשוט וישיר. מכיון ש-13 הוא מספר אי-זוגי אנו צריכים להסיר שתי לבנים החל מן הטור ה-12. כך שצריך לבדוק האם משתנה הטור הוא זוגי או לא.

תוכנית 1.9(b) מבחינה בין מספרים זוגיים ואי-זוגיים. שורה 20 מבצעת למעשה את החישוב (אם A=3 הרי ש-INT(A/2)=1 בעוד A/2=1.5), בודקת האם המספר הוא זוגי או אי-זוגי על-ידי חישוב האם המספר מתחלק בשניים.

תוכנית 1.9(a)

```
10 INPUT"NUMBER";A
20 IF INT(A/2)<>A/2 THEN PRINT"ODD":GOTO 10
30 PRINT"EVEN":GOTO 10
```

כעת, לצרוף שתי התוכניות יחדיו - זו שבונה את החומה ואחריה זו שללא הרף מקפיצה כדור, יהיה שנוי קל בגרסה הסופית, משום שהכדור יקפוץ למעלה מן התחתית רק אם המחבט יעמוד בדרכו ובתוכנית זו הוא יקפוץ אוטומטית בכל פעם שהוא מגיע לתחתית המסך, כלומר y>23.

ראשית הכנס את תוכנית הבניה, תוכנית 1.9, ואחר-כך תוכנית 1.10 והסר את שורה 80.

תוכנית 1.10

```
100 REM BOUNCING BALL
110 X=11:Y=11
120 X1=1:Y1=1
130 POKE 1024+X+Y*40,81
140 POKE 55296+X+Y*40,2
150 XX=X:YY=Y
160 IF X<1 OR X>38 THEN X1=-X1
170 IF Y<1 OR Y>23 THEN Y1=-Y1
180 X=XX+X1
190 Y=YY+Y1
200 A=PEEK(1024+X+Y*40)
210 IF A<>160 THEN 260
220 Y1=-Y1
230 IF INT(X/2)<>X/2 THEN X=X-1
240 POKE 1024+X+Y*40,32
250 POKE 1024+(X+1)+Y*40,32
255 IF X<1 THEN Y1=1:GOTO 160
260 POKE 1024+XX+YY*40,32
270 GOTO 130
```

שורה 230 מבטיחה שהלבנים יהרסו בצורה הנכונה (כלומר x הוא מספר זוגי). שורה 255 מבטיחה שאם כדור פורץ את החומה הוא קופץ לפני שהוא מגיע לשורה העליונה (זכור שהשורה העליונה נשמרת עבור פרטי התוצאות). יתרת התוכנית היא ברורה וישירה. אם ערכו של `A` הוא 160 (חומה), יש להסיר לבנה מן החומה ולבצע קפיצה. אם ערכו של 'A' שונה מ-160 הכדור ממשיך לנוע לאורך מסלולו הקודם.

כעת כשעשינו כבר את כל העבודה השחורה, זה פשוט תהליך של צרוף כל החלקים יחדיו וקבלת המוצר הסופי. השתמשנו בשלוש שגרות. אחת עבור הנעת המחבט, אחת להנעת הכדור ושגרת הפתיחה (בנית החומה, הגשת הכדור וכו'). במקום הצגת רשימות נביט ראשית בכל שגרה בנפרד.

תוכנית 1.11 היא השגרה להנעת המחבט לאורך תחתית המסך. שגרה זו נקראת פעמיים עבור כל פעם ששגרת הנעת הכדור נקראת. וכך מאפשרת למחבט לנוע מהר יותר מן הכדור. התו עבור המחבט שונה מ-160 (משבצת מלאה) ל-60 (סימן פחות מ-). דבר זה מאפשר לנו להגיב בצורה שונה בכל פעם שאנו פוגעים במחבט או בקיר.

תוכנית 1.11

```
3000 REM MOVING THE BAT
3010 GET A$:IF A$="" THEN RETURN
3020 IF A$="1" AND BC>0 THEN BC=BC-2:GOTO 3050
3030 IF A$="9" AND BC<38 THEN BC=BC+2:GOTO 3050
3040 GOTO 3060
3050 POKE 1984+CB,32:POKE 1984+CB+1,32
3060 POKE 1984+BC,60:POKE 56256+BC,0
3070 POKE 1984+BC+1,60:POKE 56256+BC+1,0
3080 CB=BC
3090 RETURN
```

הדבר הבא שיש לפתחו הוא השגרה להנעת הכדור. תוכנית (1.11(b. עם החזרה מן השגרה תערך בדיקה לערכו של NB - אם הוא 0 הרי שאין צורך בכדור חדש (והחזרה נעשית דרך שורה 2180 - דבר המצביע על תנועה חוקית). דבר זה מתאר את הרעיון החשוב של העברת אינפורמציה אל שגרות ומתוכן באמצעות דגלים אשר נקבעים לערכים מסוימים.

שים לב שאחרי שאנו מביטים (PEEK) אל הכתובת הבאה (שורה 2060) ולפני שדבר מה אחר מתבצע, נערכת בדיקה האם הכדור מתקרב למחבט. אם כן, הרי שערכו של 'A' הוא 60 והכדור נחבט למעלה - שורה 2065.

התוספת השניה לשגרה זו היא מרכיב התוצאה בשורה 2090, כאשר ערך ה-'Y' ומשמש לחישוב התוצאה. 'Y' מופחת מ-10 וערך זה מוכפל ב-5 כך שהלבנים הקרובות לפסגה שוות יותר נקודות מאשר הלבנים הנמוכות יותר. כלומר אם y=3 הרי ש-10-3=7 ו-7x5=35 כך שהסרת הלבנה .. שווה 35 נקודות.

תוכנית 1.11(b)

```
2000 REM MOVING BALL
2010 YY=Y:XX=X
2020 IF X<1 OR X>38 THEN X1=-X1
2025 IF X<0 OR X=0 THEN X1=+1
2030 IF Y<2 OR Y=2 THEN Y1=+1
2035 IF Y>23 THEN NB=1:RETURN
2040 X=XX+X1
2050 Y=YY+Y1
2060 A=PEEK(1024+X+Y*40)
2065 IF A=60 THEN Y1=-Y1:GOTO 2040
2070 IF A<>160 THEN 2140
2080 Y1=-Y1
2090 SC=SC+5*(10-Y)
2100 IF INT(X/2)<>X/2 THEN X=X-1
2110 POKE 1024+XX+YY*40,32
2120 POKE 1024+(X+1)+Y*40,32
2125 PRINT"<HOME>SCORE:";SC
2130 GOTO 2010
2140 POKE 1024+XX+YY*40,32
2150 IF Y=1 THEN Y1=1:GOTO 2010
2160 POKE 1024+X+Y*40,81
2170 POKE 55296+X+Y*40,2
2180 RETURN
```

כעת נעבור לשגרה הסופית. תוכנית 1.11(c) או למעשה שתי השגרות האחרונות כאחת. שורות 1000-1050 עוסקות בבנית החומה, מציאת אינפורמציה על התוצאה, מאפסות את התוצאה וקובעות את משתנה הכדור ל-1 (מספר הכדור המשוחק כרגע). קיימים שלושה כדורים ונערכת בדיקה סמוך לסוף השגרה לערכו של משתנה הכדור. חלק זה הכרחי רק בתחילתו של כל משחק חדש. חלקה השני של השגרה עוסק בהגשת כדור חדש וכך ש-NB נקבע לאפס (זכור שבסוף המשחק הקודם הוא נקבע לאחד לסמן שהכדור אבד) ושורות 1150 עד 1180 מבטיחות

מיקום פתיחה אקראי (משורה 11) וכיוון פתיחה אקראי. התוצאה הטובה ביותר תוסבר בחלק הבא.

תוכנית 1.11(c)

```
1000 REM INITIALISATION ROUTINE
1010 FOR X=0 TO 39
1020 FOR Y=2 TO 7
1030 POKE 1024+X+Y*40,160
1040 POKE 55296+X+Y*40,Y
1050 NEXT Y,X
1060 BC=0
1070 POKE 1984+BC,60:POKE 56256+BC,0
1080 POKE 1984+BC+1,60:POKE 56256+BC+1,0
1090 PRINT"<HOME>SCORE:";SC;TAB(12);"BAL
L:";B,"BEST:";BEST
1100 B=1
1110 SC=0
1120 REM NEW BALL SERVED
1125 POKE 1984,32:POKE 1985,32
1130 NB=0
1135 PRINT"<HOME>SCORE:";SC;TAB(12);"BALL:";B
1140 Y=11
1150 X=INT(RND(1)*36)+2
1160 Y1=1
1170 X1=INT(RND(1)*2)
1180 IF X1=0 THEN X1=-1
1190 RETURN
```

החלק הסופי - תוכנית 1.11(d) - היא התוכנית הראשית השולטת על קריאתן של השגרות.

התוכנית נעה בלולאה בין שורות 60 ל-150. בודקת ללא הרף באם נדרש כדור חדש (שורות 80 ו-130) או אם כל הלבנים נהרסו. התוצאה האפשרית המקסימלית נקבעת ל-3300 (שורה 70), ובונוס של 1000 מתווסף עבור כל כדור שנותר אחרי שכל החומה נהרסה. בסיום המשחק העכשווי - כל שלושת הכדורים בוזבזו או שכל החומה נהרסה - שורות 180 עד 220 מציגות את התוצאה ושאלות על המשחק הבא. התוצאה הטובה ביותר (`BEST`) מאופסת בתחילה ובדיקה נערכת בשורה 190 האם התוצאה העכשווית טובה יותר, ופעולה מתאימה מתבצעת.

תוכנית 1.11(d)

```
5 PRINT "<CLR>"
10 REM FINAL PROGRAM
20 POKE 53281,1
30 POKE 53280,1
40 BEST=0
50 GOSUB 1000
60 GOSUB 2000
70 IF SC=3300 THEN 170
80 IF NB=0 THEN 130
90 IF B=3 THEN 180
100 POKE 1024+X+Y*40,32
110 B=B+1
120 GOSUB 1120
130 GOSUB 3000
140 GOSUB 3000
150 GOTO 60
160 PRINT"<HOME>SCORE:";SC;TAB(12);"BALL
:";B,"BEST:";BEST
170 SC=SC+(1000+(3-B))
180 PRINT"<HOME><10DCRSR> GAME OVER"
190 IF SC>BEST THEN PRINT"NEW BEST SCORE
":BEST=SC
199 PRINT"SCORE";SC
200 PRINT" PRESS ANY KEY FOR ANOTHER GO"
205 GET A$:IF A$<>"" THEN 205
210 GET A$:IF A$="" THEN 210
220 PRINT"<CLR>":SC=0:B=0:NB=0:GOTO 50
```

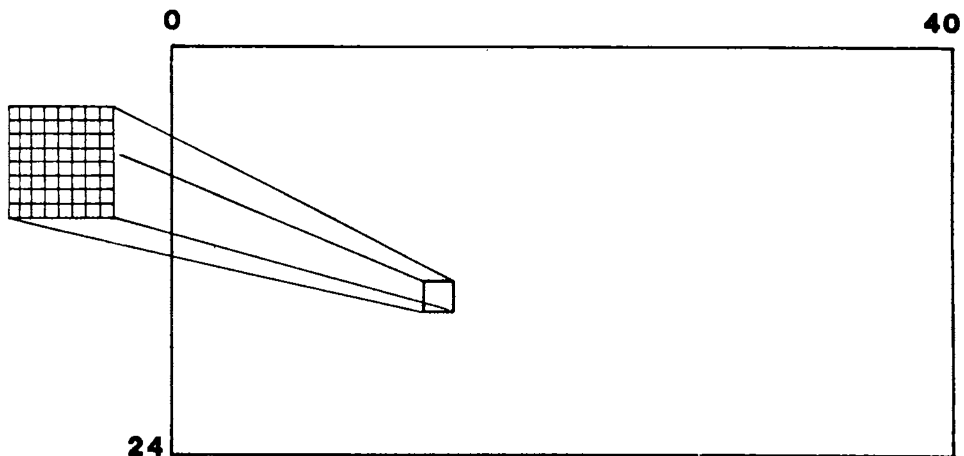
תוכנית זו הוסברה בפרטי פרטים, ואנו מקווים כי היא עניינה אותך לכל אורך הדרך, כך שעירבת כמה רעיונות משלך עבור משחקי ווידאו. למעשה קיים מספר אין-סופי של משחקים שאפשר לתכנן.

פרק 2
גרפיקה עדינה

מסך ממופה סיביות

כדי להבין כיצד פועל מסך ממופה סיביות, עליך ללמוד תחילה את הקשר בין מסך הטקסט הרגיל ומסך ממופה סיביות.

במסך טקסט, כל תו תופס תא אחד. כל תא למעשה הוא שישים וארבע (8x8) נקודות בודדות הקרויות פיקסלים (PIXELS) או סיביות (BITS). כאשר משתמשים במסך הטקסט אפשר להשתמש בנקודות אלו רק כתאים ולא כערכים נפרדים. מסך ממופה סיביות מאפשר לך בפשטות לקבוע כל סיבית בנפרד.



לכל סיבית במסך יש סיבית מתאימה בזכרון וכך שמונה סיביות שוות לבית ומכיוון שישנן 64000 סיביות נפרדות במסך ברזולוציה גבוהה הוא תופס זכרון רב.

מסך ממופה הסיביות נקבע על-ידי אחסון מספר ברגיסטר 24 של שבב VICII. שבב VICII שולט בפלט הווידאו של הקומודור והוא נמצא בכתובת זכרון 53272. רגיסטר זה שולט בדברים אחרים חוץ מהמסך ממופה הסיביות וכתוצאה מכך רק סיביות 1, 2 ו-3 של רגיסטר זה משמשות למטרה זו. כאשר אתה מדליק את הקומודור 64 סיביות אלו נקבעות לאפס.

כדי להמנע משינוי הסיביות שאינן עוסקות במיפוי סיביות, עליך תחילה לקרוא (PEEK) את תוכן 53272, להוסיף את סיביות כתובת הזכרון ולאחסן את התוצאה בכתובת 53272.
דבר זה נעשה בעזרת הפקודה הבאה:

```
POKE 53272, PEEK(53272) OR 8
```

ה'8' אומר לקומודור 64 כי המסך ממופה הסיביות יתמקם בכתובות 8192 והלאה. ה'8' מסמל 8K שהם 8x1024=8192. אחרי שקבענו היכן יהיה המשך ממופה הסיביות, השלב הבא הוא להפעיל אותו על-ידי קביעת סיבית שליטה ברגיסטר 17 של שבב המסך. רגיסטר זה נמצא בכתובת 53265. הסיבית שעלינו לקבוע היא סיבית 6 ולסיבית זו ערך עשרוני של 32.

רגיסטר השליטה שולט גם בדברים אחרים, כלומר רק סיבית 6 צריכה להשתנות. כדי לשנות רק סיבית אחת בזכרון עליך להשתמש בפעולת או (OR) של הערך שנקרא עם מספר הסיבית שצריכה להשתנות. בדוגמא זו PEEK(53265) ואחר-כך OR עם 32 (ערכה העשרוני של סיבית 6).

```
POKE 53265,PEEK(53265) OR 32
```

דבר זה שינה בהצלחה רק את סיבית 6 של כתובת הזכרון 53265 והעביר את המחשב למצב ממופה זכרון. כדי לצאת ממצב ממופה זכרון עליך פשוט לאפס את סיבית 6. דבר זה נעשה על-ידי פעולת 'וגם' (AND) של הערך בכתובת 53265 עם 223 (255-32=223).

```
POKE 53265, PEEK (53265) AND 223
```

כדי להחזיר את המסך למצב טקסט רגיל עליך לבצע פעולת AND נוספת, הפעם על כתובת 53272.
לסכום:

```
POKE 53272, PEEK (53272) OR 8
```

אומר למחשב כי המסך ממופה הסיביות ימוקם מכתובת 8192 והלאה.

```
POKE 53265, PEEK(53265) OR 32
```

מפעיל את מצב מפוי הסיביות.

```
POKE 53265, PEEK(53265) AND 223
```

מכבה את מצב מיפוי הסיביות, ולבסוף

```
POKE 53272, PEEK (53272) AND 247
```

מחזיר את המחשב למצב נורמלי.

כאשר אתה מכניס את הפקודות אשר מעבירות את הקומודור 64 למצב ממופה סיביות תבחין כי כל התוים על המסך הופכים לריבועים צבעוניים. זוהי טעות קטנה בזכרון הראם הצבעוני של הקומודור 64 אך אין מקום לדאגה. אפשר להסיר את הריבועים מעל המסך בקלות.

ניקוי המסך

כדי לנקות את המסך ממופה הסיביות כל שעליך לעשות הוא לאחסן 0 בכל כתובות מסך הסיביות. מסך הסיביות שלנו מתחיל בכתובת זכרון 8192 ומסתיים 8000 מקומות זכרון קדימה ב-16192, כך שכדי לנקות את המסך הכנס בזהירות את הפקודות הישירות הבאות:

FOR X=8192 TO 16192:POKE X,0:NEXT X

כשתלחץ RETURN תבחין שפקודה זו תסיר סיביות של צבע מן המסך אך לא תסיר את ריבועי הצבע של התוים. הסיבה לכך היא שתוים אלו נשלטים על-ידי מסך התוים ואפשר להסירן רק על-ידי אחסון ערך בכל הכתובות מ-1024 עד 2023 (כתובות הזכרון של התוים עבור מסך הטקסט). הערך שתאחסן בכתובות אלו יקבע מה תראה על המסך ברזולוציה גבוהה. כדי לשנות את המסך שיראה כחול נקי הכנס (שוב, בזהירות) את הלולאה הבאה:

FOR X=1024 TO 2023:POKE X,22:NEXT X

כעת (כשהלולאה תסתיים) המסך ממופה הסיביות הוא כחול נקי. מושלם לתחילת הציור עליו. כאשר הלולאה תסתיים תראה את ההופעה מחדש של מספר ריבועים צבעוניים. זוהי הודעת READY המופיעה על מסך הטקסט. אחרי שעשינו זאת בוא נעזוב את המסך מפוי הסיביות לרגע. הדרך המהירה ביותר לעשות זאת היא לחיצת RUN/STOP ביחד עם מקש RESTORE.

ברגע שעשינו זאת הקומודור 64 יחזיר את עצמו למצב טקסט, בצבעים שהוא משתמש כאשר הוא נדלק.

כעת נשים את הפקודות שלמדנו עד כה לתוך תוכנית.

תוכנית 2.1

```
10 REM BIT MAP PROGRAM
20 POKE 53272, PEEK(53272) OR 8
30 POKE 53265, PEEK(53265) OR 32
35 REM BIT MAP SCREEN SELECTED
40 FOR X=8192 TO 16192
50 POKE X,0:NEXT X
55 REM BIT MAP SCREEN CLEARED
60 FOR X=1024 TO 2023
70 POKE X,22:NEXT X
75 REM TEXT SCREEN CLEARED
```

צא ממצב מפוי זכרון על-ידי לחיצת RUN/STOP ו-RESTORE והוסף את השורות הבאות לתוכנית.

תוכנית 2.1(a)

```
980 GET A$:IF A$="" THEN 980
985 POKE 53265,PEEK(53265) AND 223
990 POKE 53272,PEEK(53272) AND 247
```

דבר זה ישמור את המסך עד שמקש ילחץ. כאשר ילחץ מקש, שורות 985 ו-990 יחזירו את הקומודור 64 למסך טקסט נורמלי. ברגע שאתה נמצא בחזרה במצב טקסט תראה שהמסך מלא בתוי `V`. הסיבה לכך היא שכדי לנקות את המסך ממופה הסיביות אחסנו 22 בכל כתובת טקסט, עשרים ושניים הוא ערך ה-POKE של 'V'.

כדי להפוך את התוכנית למושלמת הכנס את השורה הבאה:

תוכנית 2.1(b)

```
995 PRINT"<CLR>"
```

ציור על המסך ממופה הסיביות

כדי ליצור ציור על המסך ממופה הסיביות צריך לאחסן ערך לכל סיבית מתאימה של המסך ממופה הסיביות. כאשר ניקנו את המסך קודם לכן, הדבר נעשה על-ידי אחסנת הערך 0 בכל כתובת זכרון מ-8192 עד 16192. דבר זה קובע 8 סיביות בבת-אחת, כלומר בית (BYTE). בית אחד הוא 8 סיביות. 8192 הוא בית בזכרון ולכן הוא 8 סיביות. כדי לקבוע סיבית עלינו לחשב את ערכה יחסית למסך ובתוך הבית.

מכיון שישנם 40 בתים בכל שורת מסך קיימים גם 320 סיביות (40x8=320) בכל שורה של מסך הרזולוציה הגבוהה. חלוקת y ב-8 תמצא את מיקום הסיבית שלו בתוך הבית. התוצאה של חילוק זה כאשר היא מוכפלת ב-320 תיתן את אזור המסך עבור הסיבית. לבסוף אם תוסיף y וגם (AND) 7 תקבל בדיוק את הסיבית הנדרשת.

```
Y ציר = 8192+320*INT(y/8)+(y AND 7)
```

נניח שבחרנו את Y כ-112:

```
Y ציר = 8192+320*INT(112/8)+(112 AND 7)
  = 8192+320*14+0
  = 8192+4480
  = 12672
```

ולכן סיבית y הדרושה היא סיבית מספר 12672.

ציר x מחושב פשוט על-ידי חלוקה ב-8.

נניח כי x הוא 79:

```
x ציר = INT(79/8) = 9
```

דבר זה אומר לנו כי ה-x המבוקש הוא הבית התשיעי. כדי למצוא את הסיבית המדויקת אתה שוב מבצע AND עם 7. שני חישובים אלה יכולים להתאחד כדי ליצור את הערך המבוקש. המשוואה:

```
ערך סיבית = 8192+320*INT(y/8)+8*INT(x/8)+(y AND 7)
```

הנח כי קיימים הערכים הבאים: y=112, x=79 ובעת המשוואה נותנת:

```
ערך סיבית = 8192+320*INT(112/8)+8*INT(79/8)+(112 AND 7)
 = 8192+320*14+8*INT(9.875)+(0)
 = 8192+320*14+8*9
 = 8192+4480+72
 = 8192+4552
```

ולכן הסיבית היא 12,744.

הבעיה הבאה היא מה לאחסן ב"כתובת" זו, משום ש-12,744 אינה כתובת זכרון במובן הרגיל. זהו ערך סיבית בתוך בית. איננו יכולים בפשטות לאחסן ערך לתוכה משום שהדבר ישנה את ערך כל הסיביות בכתובת זו. במצב הרזולוציה הגבוהה לכל סיבית יש ערך זכרון של '2' בחזקת (x AND 7)-7.

כך שלפי הדוגמא שבה ל-X היה את הערך 79, החישוב נעשה כך:

```
POKE ערך = PEEK(12744) OR (2↑(7-(79 AND 7)))
= PEEK(12744) OR (2↑(7-(7))
= PEEK(12744) OR (2↑(0))
= PEEK(12744) OR 1
```

וכתוצאה מכך כדי לקבוע את הסיבית המתאימה הפקודה היא:

POKE 12744, PEEK(12744) OR **1**

המשפט הכללי לקביעת סיבית יחידה הוא:

```
POKE 8192+320*INT(Y/8)+8*INT(X/8)+(Y AND 7),PEEK(8192+320*
INT(Y/8)+8*INT(X/8)+(Y AND 7) OR (2↑(7-(X AND 7)))
```

וזה קצת מסובך. כאשר משתמשים בו בתוכנית אפשר להפריד חישוב זה לשלבים, דבר ההופך אותו קל יותר להבנה:

תוכנית 2.2

```
100 REM SETTING BIT MAP POINTS
110 REM P=POINT TO/BE SET
120 X=79:Y=112
130 P=8192+320*INT(Y/8)+8*INT(X/8)+(Y AND 7)
140 REM PV=VALUE TO BE POKED
150 PV=PEEK(P) OR (2↑(7-(X AND 7)))
160 POKE P,PV
```

כך שאם ביכולתך להריץ את התוכנית כעת תראה שכל המאמץ לא היה לשוא ובפינה השמאלית של המסך הופיעה נקודה.
כעת משלמדנו כיצד לקבוע נקודה על המסך ממופה הסיביות בואו נסחף מעט ונצייר הרבה נקודות. בעזרת פקודת RANDOM נוכל ליצור מספר קביעות אקראיות. הוסף את שורה 170 ושנה את שורה 120 כדי לקבל את תוכנית 2.3:

תוכנית 2.3

```
10 REM BIT MAP SCREEN
20 POKE 53272,PEEK(53272) OR 8
30 POKE 53265,PEEK(53265) OR 32
35 REM BIT MAPPED SCREEN SELECTED
40 FOR X=8192 TO 16192
50 POKE X,0:NEXT X
```

```
55 BIT MAPPED SCREEN CLEARED
60 FOR X=1024 TO 2023
70 POKE X,22:NEXT X
75 REM TEXT SCREEN CLEARED
100 REM SETTING BIT MAP POINTS
110 REM P=POINT TO BE SET
120 X=INT(RND(1)*320):Y=INT(RND(1)*200)
130 P=8192+320*INT(Y/8)+8*INT(X/8)+(Y AND 7)
140 REM PV=VALUE TO BE POKED
150 PV=PEEK(P) OR (2↑(7-(X AND 7)))
160 POKE P,PV
170 GET A$:IF A$="" THEN 120
980 GET A$:IF A$="" THEN 980
985 POKE 53265,PEEK(53265) AND 223
990 POKE 53272,PEEK(53272) AND 247
995 PRINT"<CLR>"
```

כעת אם תריץ את התוכנית יופיעו נקודות אקראיות על המסך. נקודות אלו ימשיכו להופיע עד אשר תלחץ על מקש. דבר זה יעצור את הנקודות (PIXELS) אך לא ישנה את המסך, כך שתוכל להביט על הדוגמא. לחיצה על מקש נוסף תחזיר אותך למצב טקסט.

ציור קוים

כדי לצייר קו עליך לדעת את הקואורדינטות של ההתחלה והסוף של הקו שלך. ראשית נעסוק בקוים מאונכים ואופקיים.
קוים אופקיים יכולים להיווצר על-ידי הגבלת קואורדינטת x.
כלומר: עבור שורה המתחילה ב-(10,100) ומסתיימת ב-(310,100) פקודות בייסיק (BASIC):

```
120 Y=100:FOR X=10 TO 310
165 NEXT X
```

כשתוספנה לתוכנית 2.3 תשרטטנה קו לאורך אמצע המסך. הקו זז לאט לאורך המסך, סיבית אחר סיבית.
שורות אופקיות משורטטות ע"י הגדלת ציר Y, כלומר:
לשורה שמתחילה ב-(160,10) ונגמרת ב-(160,190) פקודות בייסיק:

```
120 X=160:FOR Y=10 TO 190
165 NEXT Y
```

כאשר יתווספו לתוכנית 2.3 יציירו קו אופקי באמצע המסך!
בשיטה דומה אפשר לשרטט גם קוים אנכיים.
בעזרת שילוב של שגרות ציור קוים אנכיים ואופקיים יהיה אפשר לצייר ריבוע. מה שידרש הם ערכי x ו-y וערך גודל הריבוע - SS. כדי לצייר ריבוע אשר הקואורדינטות שלו וגודלו יוכתבו על-ידי המשתמש הכנס את תוכנית 2.4.

תוכנית 2.4

```
10 INPUT"X START COORDINATE";X
20 INPUT"Y START COORDINATE";Y
30 INPUT"SQUARE SIZE";SS
40 POKE 53272,PEEK(53272) OR 8
50 POKE 53265,PEEK(53265) OR 32
60 FOR L=8192 TO 16192
70 POKE L,0:NEXT L
80 FOR L=1024 TO 2023
90 POKE L,22:NEXT L
100 REM DRAW THE SQUARE
110 FOR X=X TO X+SS :GOSUB 1000
120 POKE P,PV:NEXT X
130 FOR Y=Y TO Y+SS:GOSUB 1000
140 POKE P,PV:NEXT Y
150 FOR X=X TO X-SS STEP-1:GOSUB 1000
160 POKE P,PV:NEXT X
170 FOR Y=Y TO Y-SS STEP-1:GOSUB 1000
180 POKE P,PV:NEXT Y
190 GET A$:IF A$="" THEN 190
200 POKE 53265,PEEK(53265) AND 223
210 POKE 53272,PEEK(53272) AND 247
215 PRINT"<CLR>"
220 END
1000 P=8192+320*INT(Y/8)+8*INT(X/8)+(Y AND 7)
1010 PV=PEEK(P) OR (2↑(7-(X AND 7)))
1020 RETURN
```

שורות 100 עד 180 הן אלו שלמעשה מציירות את הריבוע. שורות 110 ו-120 מציירות את הקו האופקי הראשון המתחיל בקואורדינטות (x,y) ומסתיים ב-(x+SS,y). כאשר לולאה זו מסתיימת ערכו של x הוא x+SS. אם ערך x אותו בחרת היה 100 וגודל הריבוע היה 50, הרי שאחרי הלולאה ערכו של x הוא 150.

כך שכאשר מצויר הקו האופקי השני הלולאה הולכת מ-x אל SS-x (או בדוגמא למעלה מ-150 ל-50-150) ואותם החוקים תופסים גם לגבי הקוים האנכיים.

קוים אלכסונים

ציור קוים אלכסוניים קשה הרבה יותר מכיוון שגם ציר x וגם ציר y צריכים להשתנות. לצייר קו אלכסוני כאשר הזוית היא 45 מעלות אין בעיה, פשוט מגדילים את ציר x ו-y בצורה שווה. דבר זה מודגם בתוכנית 2.5 אשר מציירת קו מהכתובת השמאלית העליונה בכיוון מטה אל התחתית הימנית.

תוכנית 2.5

```
10 POKE 53272,PEEK(53272) OR 8
20 POKE 53265,PEEK(53265) OR 32
30 FOR L=8192 TO 16192
40 POKE L,0:NEXT L
50 FOR L=1024 TO 2023
60 POKE L,22:NEXT L
70 FOR X=0 TO 199:Y=X
80 GOSUB 1000:POKE P,PV:NEXT X
90 GET A$:IF A$=""THEN 90
100 POKE 53265,PEEK(53265) AND 223
110 POKE 53272,PEEK(53272) AMD 247
120 PRINT"<CLR>"
130 END
1000 REM CALCULATE BIT
1010 P=8192+320*INT(Y/8)+8*INT(X/8)+(Y AND 7)
1020 PV=PEEK(P) OR (2↑(7-(X AND 7)))
1030 RETURN
```

שרטוט קוים אלכסוניים שאינם בזוית 45 מעלות, אינו כה קל - כדי להדגים כיצד זה נעשה נשתמש בקואורדינטות לדוגמא:

מ-(20,20) ל-(160,100)

מ-(X1,Y1) ל-(X2,Y2)

הצעד הראשון הוא למצוא האם הקו האלכסוני יהיה יותר לרוחב מאשר למטה או יותר למטה מאשר לרוחב. דבר זה נעשה על-ידי השוואת הערך

x2-x1 עם y2-y1. הערך המוחלט של כל תוצאה נלקח למקרה שהקואורדינטות הן כאלו שיצריכו ציור לשמאל, כלומר y1=100,x1=160 ו-y2=20,x2=20-.

החישוב עד כה:

```
ABS (X2-X1)-ABS(Y2-Y1)=?
ABS(160-20)-ABS(100-20)=?
140  -  80      =60
```

מכיון שהתוצאה של חישוב זה היא חיובית הקו יזוז מטה, יותר בכיוון x מאשר y.

הקו מ-(x1,y1) ל-(x2,y2) יצויר ב-140 שלבים. 80 יהיו בכיוון y וה-60 הנותרים יהיו בכיוון x.

כדי ליצור קו אלכסוני חלק יש צורך להפריד את השלבים האלכסוניים מתחילה ועד הסוף.

דבר זה נעשה על-ידי שמוש במשתנה סכום הריצה (RT) אשר בתחילה ניתן לו הערך של מחצית ההבדל הגדול ביותר בין x או y. במקרה זה, הבדל ה-x גדול יותר וכך:

```
RT=INT(140/2)
RT=70
```

בכל מעבר בלולאת ציור הקו, RT יגדל בערך הנמוך של מבחן ה-x2-x1 וה-y2-y1, בדוגמא זו 80.

לכן RT=RT+80

ערך זה נבחן כעת מול 140 (x2-x1). אם RT גדול, הרי שיופחת 140 מ-RT והנקודה תקבע בציר y מזיזה את הקו מטה והצידה סיבית אחת. אם RT אינו גדול מ-140 הרי שהנקודה נקבעת בציר x והקו יזוז הצידה. תהליך זה נמשך עד שהלולאה מסתיימת.

החישוב במלואו הוא:

```
1) XX=ABS(X2-X1):YY=ABS(Y2-Y1)
IF XX>YY THEN LL=XX:DD=YY
IF YY>XX THEN LL=YY:DD=XX
IF XX=YY THEN LL=XX:DD=YY
```

דבר זה בודק האם הקו האלכסוני גדול יותר אופקית (LL=xx:DD=yy) או אנכית (LL=yy:DD=xx). אם xx ו-yy שוים הרי שהקו הוא ב-45 ואין זה משנה באיזו דרך יקבעו DD ו-LL.

```
2) SX=SGN(XX):SY=SGN(YY)
```

דבר זה יגיד לנו האם על הקו להיות מצויר שמאלה ולמעלה (Sx=-1,Sy=-1), שמאלה ומטה (Sx=-1,Sy=1), ימינה ומטה (Sx=1,Sy=1) או ימינה ומעלה (Sx=1,Sy=-1). בדוגמא שאנו משתמשים Sx ו-Sy הם 1 והקו נע ימינה ומטה.

```
3) RT=INT(LL/2)
FOR L=1 TO LL
RT=RT+DD
IF RT<LL THEN X=X+SX:GOSUB 1000
IF RT>LL THEN RT=RT-LL:Y=Y+SY:X=X+SX:GOSUB 1000
```

כפי שהוזכר קודם, קטע זה מחליט האם קביעת הסיבית תהיה אופקית או אנכית. אם RT קטן מ-LL היא אופקית, אך אם RT גדול מ-LL הרי הסיבית הבאה תקבע אנכית.

השגרה בשורה 1000 והלאה היא זו שמחשבת את הסיבית לקביעה. התוכנית לציור קו בכל כיוון (אפשר לצייר מעלה ומטה בשיטה זו) רשומה בתוכנית 2.6.

תוכנית 2.6

```
10 REM LINE DRAWING PROGRAM
20 INPUT"X START COORDINATE";X1
30 INPUT"Y START COORDINATE";Y1
40 INPUT"<DCRSR>X STOP COORDINATE";X2
50 INPUT"Y STOP COORDINATE";Y2
60 POKE 53272,PEEK(53272) OR 8
70 POKE 53265,PEEK(53265) OR 32
80 FOR L=8192 TO 16192
90 POKE L,0:NEXT L
100 FOR L=1024 TO 2023
110 POKE L,22:NEXT L
120 XX=ABS(X2-X1):YY=ABS(Y2-Y1)
130 SX=SGN(XX):SY=SGN(YY):X=X1:Y=Y1
140 IF XX>YY THEN LL=XX:DD=YY
150 IF XX<YY THEN LL=YY:DD=XX
160 IF XX=YY THEN LL=XX:DD=YY
170 RT=INT(LL/2)
180 FOR L=1 TO LL
190 RT=RT+DD
200 IF RT<LL THEN X=X+SX:GOSUB 1000
```

```
210 IF RT>LL THEN RT=RT-LL:Y=Y+SY:X=X+SX
:GOSUB 1000
220 POKE P,PV:NEXT L
230 GET A$:IF A$="" THEN 230
240 POKE 53265,PEEK(53265) AND 223
250 POKE 53272,PEEK(53272) AND 247
260 PRINT"<CLR>":END

1000 REM CALCULATE THE BIT
1010 P=8192+320*INT(Y/8)+8*INT(X/8)+(Y AND 7)
1020 PV=PEEK(P) OR (2↑(7-(X AND 7)))
1030 RETURN
```

תוכנית זו תצייר רק קו אחד ואז תבצע END. קל למדי לשנות את התוכנית כך שסדרה של קוים אקראיים יצוירו, בצורה די דומה לדרך שבה הופקו נקודות אקראיות.

תוכנית 2.7 היא תוכנית ציור קוים אקראיים ותעצור כאשר ילחץ מקש הרווח (SPACE BAR). לחיצה שניה על מקש הרווח תחזיר אותך למסך הטקסט.

תוכנית 2.7

```
10 REM RANDOM LINE DRAWING
20 POKE 53272,PEEK(53272) OR 8
30 POKE 53265,PEEK(53265) OR 32
40 FOR L=8192 TO 16192
50 POKE L,0:NEXT L
60 FOR L=1024 TO 2023
70 POKE L,22:NEXT L
80 X1=INT(RND(1)*320):X2=INT(RND(1)*320)
90 Y1=INT(RND(1)*200):Y2=INT(RND(1)*200)
100 XX=ABS(X2-X1):YY=ABS(Y2-Y1)
110 SX=SGN(XX):SY=SGN(YY):X=X1:Y=Y1
120 IF XX>YY THEN LL=XX:DD=YY
130 IF XX<YY THEN LL=YY:DD=XX
140 IF XX=YY THEN LL=YY:DD=XX
150 RT=INT(LL/2)
160 FOR L=1 TO LL
170 RT=RT+DD
180 IF RT<LL THEN X=X+SX:GOSUB 1000
190 IF RT>LL THEN RT=RT-LL:Y=Y+SY
:X=X+SX:GOSUB 1000
200 POKE P,PV:NEXT L
```

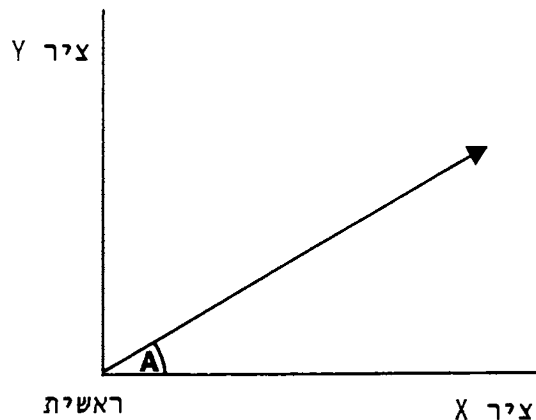
```
210 GET A$:IF A$<>" " THEN 80
220 GET A$:IF A$=" " THEN 220
230 GET A$:IF A$<>" " THEN 230
240 POKE 53265,PEEK(53265) AND 223
250 POKE 53272,PEEK(53272) AND 247
260 PRINT"<CLS>":END

1000 REM CALCULATE THE BIT
1010 P=8192+320*(Y/8)+8*INT(X/8)+(Y AND 7)
1020 PV=PEEK(P) OR (2↑(7-(X AND 7)))
1030 RETURN
```

ציור מעגלים

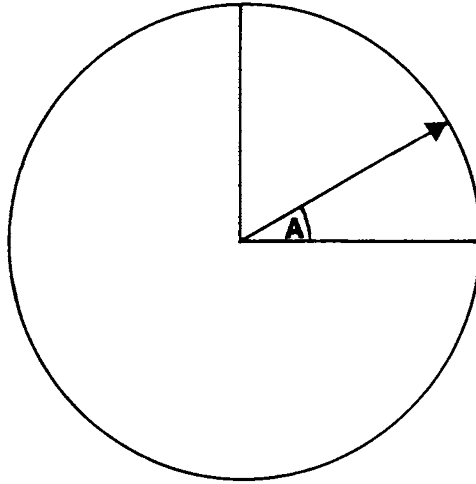
כדי לצייר מעגל עלינו להשתמש ב-SIN וב-COS. אם אינך שולט עדיין במושגים אלה, תוכל להעזר בסידרת "מחשבת" ללימוד הקומודור 64.

ציור 2.2 מראה זוג צירים ומיתר באורך יחידה אחת מראשית הצירים ובזוית 'A' מציר x.



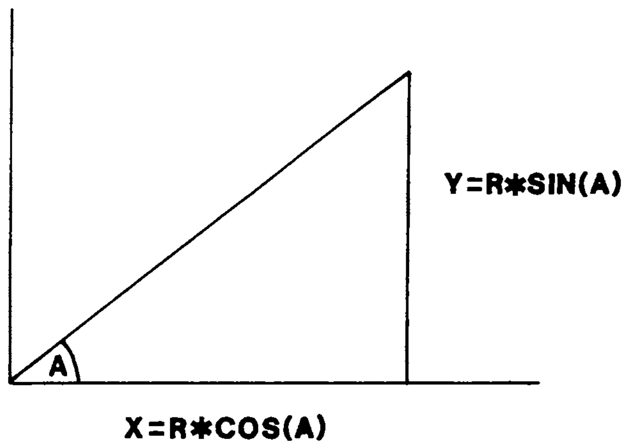
ציור 2.2

אם מיתר זה יסתובב סביב הראשית, קצהו ישאיר אחריו מעגל:



כדי להשתמש ברעיונות אלו לציור מעגל על המסך ממופה הסיביות, נצטרך לדעת את קואורדינטות x ו-y של נקודות הקצה של המיתר. כאן משתמשים ב-SIN ו-COS.

עבור זוית מסוימת 'A' קואורדינטות x ו-y הן:



ציור 2.3

הקומודור 64 עובד ברדיאנים. ישנם 2*Pi רדיאנים ב-360 מעלות (מעגל). Pi הוא היחס בין רדיוס המעגל להיקפו. היקף המעגל הוא: רדיוס*2*Pi. הערך האמיתי של Pi הוא 3.14159. לקומודור 64 מקש Pi, המסומן π. סמל זה ישמש מכאן והלאה לציון Pi.

אם A שווה לכל הזויות במעגל (0π ל-2π) הרי ש-(SIN (A ו-COS(A) נותנים את כל קואורדינטות x ו-y של הנקודות סביב המעגל.

ליתר דיוק, קואורדינטות x ו-y הן:

X=R*COS(A):Y=R*SIN(A)

הלולאה לציור מעגל תהיה מ-0 ל-2π.

תוכנית 2.8

```
10 REM CIRCLE DRAWING
20 POKE 53272,PEEK(53272) OR 8
30 POKE 53265,PEEK(53265) OR 32
40 FOR L=8192 TO 16192
50 POKE L,0:NEXT L
60 FOR L=1024 TO 2023
70 POKE L,22:NEXT L
80 R=100:REM R=RADIUS
90 FOR A=0 TO 2*π
100 Y=R*SIN(A)+100:X=R*COS(A)+100
110 GOSUB 1000:POKE P,PV
120 NEXT A
130 GET A$:IF A$="" THEN 130
140 POKE 53265,PEEK(53265) AND 223
150 POKE 53272,PEEK(53272) AND 247
160 PRINT"<CLR>":END

1000 REM CALCULATE THE BIT
1010 P=8192+320*INT(Y/8)+8*INT(X/8)
+(Y AND 7)
1020 PV=PEEK(P) OR (2↑(7-(X AND 7)))
1030 RETURN
```

כאשר תוכנית 2.8 מורצת, כל שמוצג על המסך הן כמה נקודות. הצעד (STEP)1 שהוא ברירת מחדל, גדול מדי מכדי לקבל מעגל שלם. אנו זקוקים לגודל צעד קטן בהרבה. שנה את שורה 90 לשורה הבאה:

```
90 FOR A=0 TO 2*πSTEP 0.01
```

גודל צעד זה הוא מושלם ואם תריץ את התוכנית שנית תראה מעגל מושלם.

```
10 REM RANDOM CIRCLES
20 POKE 53272,PEEK(53272) OR 8
30 POKR 53265,PEEK(53265) OR 32
40 FOR L=8192 TO 16192
50 POKE L,0:NEXT L
60 FOR L=1024 TO 2023
70 POKE L,22:NEXT L
80 R=INT(RND(1)*50)
90 C=INT(RND(1)*320):C2=INT(RND(1)*200)
100 FOR A=0 TO 2*πSTEP 0.01
110 X=C+R*COS(A):Y=C2+R*SIN(A)
120 GOSUB 1000:POKE P,PV
130 NEXT A
140 GET A$:IF A$=""THEN 80
150 GET A$:IF A$<>""THEN 150
160 GET A$:IF A$<>" "THEN 160
170 POKE 53265,PEEK(53265) AND 223
180 POKE 53272,PEEK(53272) AND 247
190 PRINT "<CLR>":END

1000 REM CALCULATE THE BIT
1010 P=8192+320*(Y/8)+8*INT(X/8)+(Y AND 7)
1020 PV=PEEK(P) OR (2↑(7-(X AND 7)))
1030 RETURN
```

כדי לעצור את התוכנית לחץ על מקש הרווח. כדי לחזור למסך הטקסט לחץ שנית על מקש הרווח.

מיפוי מסך רב צבעוני (MULTICOLOUR BIT MAP)

עד כה כל מה שצויר על המסך ממופה הסיביות היה באותו הצבע. אפשר לקבל ארבעה צבעים שונים על המסך ממופה הסיביות. ברירה זו נבחרת על-ידי שינוי סיבית חמש בכתובת 53270.

POKE 53270,PEEK(53270) OR 16

כאשר משפט זה מבוצע, בהשוואה למצב מפוי סיביות, שנוי קטן חל בנקודות שעל המסך. כעת כאשר סיבית נקבעת, נקבעת גם הסיבית שלידה.

בנוסף למעבר למצב רב-צבעוני יש צורך לחשב את צבע הסיבית. דבר זה נעשה בפשטות בשני שלבים.

ראשית עליך לבחור את הצבע שלך. המבחר הוא:

שחור 0

לבן 1

אדום 2

ירוק בהיר 3

כדי לבחור בלבן אתה פשוט נותן ערך 1 למשתנה. כלומר

C=1

צבע אקראי נקבע בדרך הבאה:

C=INT(RND(1)*4)

הצעד הבא הוא לשנות את הצבע (C) על-ידי הבטוי

(2↑(6-(X 7)))) AND 7)))

PV=PEEK(P) OR (C≠(2↑(6-(X AND

חלק ה-(7-(X AND 7)) שונה ל-(6-(X AND 7)) מכיון שכל סיבית שנקבעת קובעת גם את שכנתה ובכך מגבילה את מפוי המסך ל-160 סיביות נפרדות. חשוב לשים לב כי ציר Y אינו מושפע מכך.

כדי להדגים את המצב הרב-צבעוני ואת המסך בן 160 הסיביות הנפרדות הכנס את תוכנית 2.10.

תוכנית 2.10

```
10 REM MULTICOLOUR BIT MAP MODE
20 POKE 53272,PEEK(53272) OR 8
30 POKE 53265,PEEK(53265) OR 32
40 FOR L=8192 TO 16192
50 POKE L,0:NEXT L
60 FOR L=1024 TO 2023
70 POKE L,22:NEXT L
80 POKE 53270,PEEK(53270) OR 16
90 FOR X=0 TO 319 STEP 2
100 Y=100:C=INT(RND(1)*4)
110 GOSUB 1000:POKE P,PV
115 NEXT X
120 GET A$:IF A$<>" "THEN 90
130 POKE 53270,PEEK(53270) AND 239
140 POKE 53265,PEEK(53265) AND 223
150 POKE 53272,PEEK(53272) AND 247
160 PRINT"<CLR>":END
1000 REM CALCULATE BIT
1010 P=8192 +320*INT(Y/8)+8*INT(X/8)+(Y AND 7)
1020 PV=PEEK(P) OR (C*(2↑(6-(X AND 7))))
1030 RETURN
```

כעת ביכולתך להשתמש במסך ממופה הסיביות של הקומודור 64 על כל אפשרויותיו. בהצלחה!

מצבים רב־צבעוניים

תווים רב־צבעוניים

רק כרגע הראינו כי קיים מצב מפוי סיביות רב-צבעוני אשר נקבע על-ידי אחסון ב-53270. כאשר דבר זה נעשה בזמן שהיינו במצב מפוי סיביות נוצר מצב מפוי סיביות צבעוני.

בכל אופן, אם:

POKE 53270,PEEK(53270) OR 16

מוכנס מחוץ למצב מפוי סיביות הרי שמצב תוים רב-צבעוניים נכנס לפעולה. במצב זה אפשר לקבל ארבעה צבעים לכל תו.

מתוך ארבעת הצבעים האפשריים, שניים הם הצבעים הנוכחיים של התוים והצבע של הרקע (נשלט על-ידי כתובת 53281). הצבעים האחרים נשלטים על-ידי כתובות 53282 ו-53283.

כדי להדגים את מצב התוים הרב-צבעוניים הכנס את התוכנית הבאה.

תוכנית 3.1

```
10 REM MULTICOLOUR CHARACTER MODE
20 PRINT"<CLR>":POKE 53281,15
30 FOR X=0 TO 39
40 POKE 1664+X,83:POKE 55936+X,8
50 NEXT X
60 POKE 53270,PEEK(53270) OR 16
70 FOR C1=0 TO 7:PRINT TAB(5*C1)"<BLK>";C1
80 FOR C2=0 TO 7:PRINT TAB(5*C1)"<WHT>";C2
90 POKE 53282,C1:POKE 53283,C2
100 FOR D=1 TO 500:NEXT D
110 NEXT C2:PRINT"<HOME>":NEXT C1
120 POKE 53270,PEEK(53270) AND 239
```

התוכנית מסתיימת בהחזרת המחשב למצב תוים נורמלי. דבר זה נעשה על-ידי AND בין תוכן כתובת 53270 ו-239.

POKE 53270,PEEK(53270) AND 239

כפי שראינו, שבב ה-Vic שולט במסך ממופה הסיביות ואז יש לו גם יתרונות אחרים.

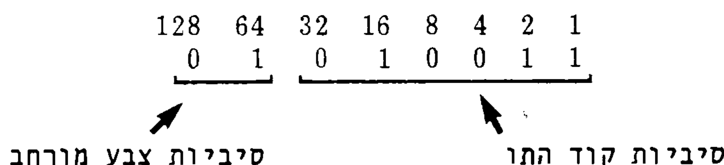
על-ידי שינוי סיבית שש של שבב ה-Vic ל-'1' (באופן רגיל הוא אפס) הקומודור 64 מפעיל את מצב צבע רקע מורחב. מצב צבע רקע מורחב מאפשר למשתמש לקבל עד ארבעה צבעים שונים על המסך.

סיבית שש של שבב ה-Vic נקבעת על-ידי אחסון תוכנו של 53265 בפעולת "או" (OR) עם 64 ב-53265:

כלומר: **POKE 53265,PEEK(53265) OR 64**

אם תכניס זאת כפקודה בהכנסה ישירה, לא תראה הרבה שנוי. כעת החזק את SHIFT ולחץ על כל אחד ממקשי האותיות ותראה שתקבל תו מקש נגטיבי ולא את סימן ה-SHIFT.

כאשר מצב הצבע המורחב נמצא בפעולה המשתמש מוגבל לשישים וארבעה תוים בלבד, 0 עד 63. אלו הם ערכי תוי מסך ולא ASCII. מצב צבע מורחב משתמש בשתי הסיביות היותר משמעותיות של כל בית של תו. דבר זה יכול להראות בדרך הבאה:



שרטוט זה מראה את ערך הבית עבור 'S' במצב צבע מורחב. שש סיביות קוד התו מצטרפות ל-19 שהוא קוד המסך עבור 'S'. שתי הסיביות היותר משמעותיות מצטרפות ל-64 שהוא קוד הצבע של לבן. בדוק זאת בעצמך על-ידי הכנסת השורה הבאה:

```
PRINT"<CLR>S"=and then press RETURN
PRINT PEEK(1024)=and then press RETURN
```

מקש ה-CLR כאשר ילחץ יראה כ-'S' הפוך. אל תדאג לכך כעת. המסך צריך להראות כך:

```
S

READY
PRINT PEEK(1024)
19

READY
```

ערך ה-PEEK של 'S' הוא 19.

כעת העבר את הסמן לנקודת הבית (HOME) ועם החזקת SHIFT לחץ על 'S'. כעת 'S' הפוך יופיע על המסך. אם כעת תקרא את תוכן כתובת 1024 תראה שהיא מכילה 83. זהו ערך הצבע '64' שנוסף לערך התו 19.

למרות שתוי ה-SHIFT נראים הפוכים הם בעצם תוים ריאליים עם ערך רקע עצמאי של לבן.

כבוי מצב צבע מורחב נעשה על-ידי AND בין תוכן כתובת 53265 עם 191 כלומר:

POKE 53265,PEEK (53265) AND 191

ביחד עם בחירת מצב צבע מורחב יכול המשתמש לבחור איזה צבע רקע יכיל כל תו. הבחירה מוגבלת למדי, קיימים רק ארבעה צבעים אפשריים והם:

כתובת זכרון 53281	0_{10}	00_2
כתובת זכרון 53282	1_{10}	01_2
כתובת זכרון 53283	2_{10}	10_2
כתובת זכרון 53284	3_{10}	11_2

כשנקבעות הסיביות הן לא מקבלות ערך של צבע אלא אילו תוים ישתנו.

00_2 - משנה את כל התוים בין הערכים 0 ו-63 ונשלט על-ידי רגיסטר 53281.

01_2 - משנה את כל התוים בין הערכים 64 ו-127 ונשלט על-ידי רגיסטר 53282.

10_2 - משנה את כל התוים בין הערכים 128 ו-191 ונשלט על-ידי רגיסטר 53283.

11_2 - משנה את כל התוים בין הערכים 192 ו-255.

מכיון שמצב תוים רב-צבעוני ומצב רקע צבעוני משתמשים שניהם בכתובות 53281, 53282 ו-53283 עליך לוודא כי רק אחד מן המצבים יבחר.

כדי להדגים את מצב הרקע המורחב הרץ את התוכנית הבאה:

תוכנית 3.2

```
10 POKE 53265,PEEK(53265) OR 64
20 FOR X=0 TO 63
30 POKE 1024+X,X:POKE 55296+X,0
40 NEXT X
50 FOR X=64 TO 127
60 POKE 1124+X,X:POKE 55396+X,0
70 NEXT X
80 POKE 53282,1
90 FOR X=128 TO 191
100 POKE 1224+X,X:POKE 55496+X,0
110 NEXT X
120 POKE 53283,5
130 FOR X=192 TO 255
140 POKE 1324+X,X:POKE 55596+X,2
150 NEXT X
160 POKE 53284,7
170 POKE 53281,15
```

התוכנית מאחסנת ארבע קבוצות נפרדות של תוים. הקבוצה הראשונה הם תוי מסך בין 0 ל-63 והאחרונה 192-255. ארבעת הקבוצות מכילות את אותם התוים משום שמצב רקע מורחב מגביל את המשתמש לתוים אלו בלבד.

זה אחר זה, כל קבוצת תוים מאוחסנת על המסך וצבע הרקע שלה נקבע. כתובת צבע הרקע 53281 אינה מאוחסנת עד הסיום כדי להראות שכתובות הרקע האחרות, העכשויות, לא יקבעו אך כל כתובות המסך האחרות - כן. כל כתובות המסך הריקות הן בעלות קוד מסך 32, רווח, אחסון ב-53281 משנה את כל כתובות המסך הריקות. כל קבוצת תוים מכילה את התוים הבאים:

```
abcdefghijklmnopqrstuvwxyz[#]  !
"#$%&'()*+,-./0123456789:;<=>?
```

ואלו הם התוים היחידים שתוכל להשתמש בהם. ודאי הבנת את זה עד כה.

פרק 3
עורך סימנים

חלק 1
גרפיקה המוגדרת על ידי המשתמש

על-ידי שמוש בגרפיקה המוגדרת על-ידי המשתמש בקומודור 64 אפשר להגדיר תו המעוצב על-ידך ואחר-כך לקרוא לו מתוכניות כאשר יש בו צורך. אחרי שהוגדר, אפשר להתנהג לתו זה בדיוק כמו לכל אחד אחר והוא עונה לשם אשר נתת לו, שם שיכול להיות כל אחד מן התוים על המקלדת או בכל מקום בקבוצת התוים. התו אותו אתה מגדיר מורכב ממטריצה של 8*8, של ריבועים קטנים הידועים בשם פיקסלים (PIXELS). אלו נקבעים למלא או ריק על-ידי התיחסות לתכניות השמורות במידע התוים השמור בזכרון ה-ROM. המחשב יכול למעשה לשמור את המידע כסדרה של 64 כתובות זכרון נפרדות אשר כל אחת מהן נקבעת ל"מלא" או "ריק" ובכך להגדיר מטריצה כמו ציור 3.1.

0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0
1	1	1	1	1	1	1	1
0	0	0	1	1	0	0	0
0	1	1	1	1	1	1	0
0	1	0	0	0	0	1	0
1	1	0	0	0	0	1	1

0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0
1	1	1	1	1	1	1	1
0	0	0	1	1	0	0	0
0	1	1	1	1	1	1	0
0	1	0	0	0	0	1	0
1	1	0	0	0	0	1	1

ציור 3.1

הגדרת תו על-ידי שמוש בבתים נפרדים עבור כל פיקסל תבזבז 64 כתובות זכרון עבור כל תו ובזמן שכמה תוים כבר אוחסנו, הם יתחילו לבלוע הרבה מזכרון המחשב. בכל אופן, העזרה נמצאת בשיטה ששבב המחשב מאחסן מידע באופן רגיל.

שבב המיקרופרוססור אשר עליו מבוסס הקומודור 64, שבב ה-6510, נחשב כשבב 8 סיביות, אשר משמעותו היא שכל תו מאוחסן במחשב בשמונה סיביות של מידע. קבוצה זו של שמונה סיביות ידועה כבית אחד. מכיוון שהשבב הוא מיקרופרוססור דיגיטלי, כל סיבית מידע יכולה להקבע רק ל"פתוח" או "סגור", כלומר '1' או '0'. כתוצאה מכך רק בית אחד דרוש לאחסנת כל שורה עבור כל אחת מצורות התוים. כך שסך הכול שמונה בתים דרושים להחזקת צורה שלימה של כל אחד מתוי ה-64. שיטת הדחיסה של שמונה סיביות של מידע תוים לתוך בית פשוט מנצלת את השיטה הבינארית. כאשר משתמשים בטכניקה זו צריך לחבר את כל האלמנטים של התו סיבית אחר סיבית, שיתנו מספר שיאוחסן בזכרון. כתוצאה מכך, לאחסון השורה העליונה של אדם רוקד הנראה בציור 3.1, פעולת החיבור מבוצעת כמו בציור 3.2.

(0x128)+	(0x64)+	(1x32)+	(1x16)+	(1x8)+	(1x4)+	(0x2)+	(0x1)	=60
0	0	1	1	1	1	0	0	$=00111100_2$

ציור 3.2

מציור 3.2 רואים כי מרכיבי השורה מסתכמים ב-60=32+16+8+4, כלומר המספר 60 צריך להיות מאוחסן בזכרון. כדי להשיג ערכי POKE עבור כל הצורה, צריך לחשב כל שורה בנפרד, כפי שנראה בציור 3.3.

בינארי

128	64	32	16	8	4	2	1	חישוב	עשרוני
0	0	1	1	1	1	0	0	32+16+8+4	= 60
0	0	1	1	1	1	0	0	32+16+8+4	= 60
0	0	0	1	1	0	0	0	16+8	= 24
1	1	1	1	1	1	1	1	128+64+32+16+8+4+2+1	= 255
0	0	0	1	1	0	0	0	16+8	= 24
0	1	1	1	1	1	1	0	64+32+16+8+4+2	= 126
0	1	0	0	0	0	1	0	64+2	= 66
1	1	0	0	0	0	1	1	128+64+2+1	= 195

ציור 3.3

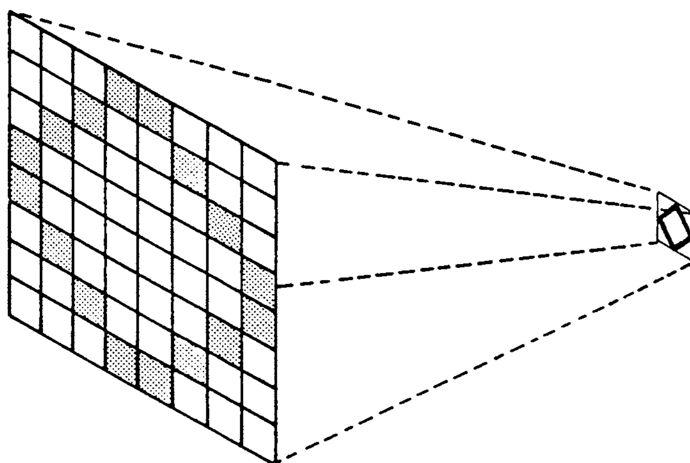
אחרי שחושבו הערכים הדרושים להגדרת תו, ערכים אלו חייבים להיות מאוחסנים באיזור הזכרון אשר שם מחפש שבב ה-VicII את התוים שלו.

מכיון שהדבר הוא תהליך מסובך למדי, קל יותר להשתמש בתוכנית שירות (UTILITY). חלק שניים של פרק זה מפתח תוכנית כזו ובזמן תהליך הפיתוח יחקר כל תהליך הגדרת התוים.

חלק 2

תוכנית שרות גרפית: הגדרה

תוכנית שרות זו תספק אמצעי ליצירת תוי גרפיקה המוגדרים על-ידי המשתמש. התוים יעוצבו על "מסך" או מטריצה אשר תייצג תו יחיד בהגדלה של פי שמונה, ראה ציור 3.4, והצורה האמיתית של התו תוצג במהלך תהליך העיצוב. על-ידי הזזת ריבוע, אשר מייצג פיקסל יחיד, מסביב למטריצת התו ואחסונו בכל מקום בו נרצה, יבנה תו, ערכי ה-POKE שלו יחושבו ויאוחסנו ב-RAM. יפותח אמצעי לאחסון ערכי POKE אלו במשפטי DATA לצורך איחוד עם תוכניות המשתמש. כדי לאפשר עבודה בתוכנית השרות יחד עם תוכניתו של המשתמש, תמוספר תוכנית השרות הגרפית משורה 60000 ואילך.



ציור 3.4

מיקום מחדש של קבוצת התוים

הקומודור 64 בא עם קבוצת תוים שלו המוכנה לשימוש. אלו הם התוים אשר אתה רואה על המסך כאשר אתה לוחץ על המקשים. אלו, בכל אופן, נשמרים ב-ROM, או זכרון קריאה בלבד READ ONLY MEMORY ואין אפשרות לשנות אותם. וכך, אם ברצונך לעצב קבוצת תוים חדשה ראשית כל יש צורך להעביר את קבוצת התוים הקיימת לתוך ה-RAM שם תוכל להתעסק בה. מכיוון שכל תו תופס שמונה בתים וישנם 256 תוים שונים בקבוצת התוים הגרפיים העליונים (UPPER CASE) להעברה, צריך למקם מחדש 2048 או 8*256 בתים. ב-ROM התוים הרגילים ממוקמים מכתובת זכרון 53248 ($D000) ואילך ומקום מומלץ אחד להעבירם אליו הוא כתובת הזכרון 12288 ($3000). שגרת PEEK/POKE ישירה למדי נראית כיכולה לבצע פעולה זו. כלומר:

```
FOR I=0 TO 2047:POKE(12288+I),PEEK(53248+I):NEXT
```

אך בגלל הדרך שבה השבבים השונים מתיחסים זה לזה, יש צורך לעשות כמה דברים אחרים לפני ביצוע תהליך ההעברה.

הגנת תקרת הזכרון

אחרי שדבר זה נעשה, קיים העתק של קבוצת התוים בפסגת ה-RAM. כמובן, די ברור שהוא ימחק על-ידי כל תוכנית שתנסה להשתמש באותו חלק מן ה-RAM. בעיה זו יכולה להפתר על-ידי "שכנוע" המחשב כי ה-RAM מסתיים יותר נמוך מאשר הוא מסתיים באמת. בשעת ההדלקה, המחשב בודק את ה-RAM ומחשב בדיוק כמה זכרון נמצא בהישג-יד. כאשר החישוב נעשה, מיקום תקרת הזכרון מאוחסן בכתובות זכרון 55 ו-56. אם כתובות אלו ישוכנו ויקבעו למספר נמוך יותר, הבייסיק של ה-64 יחשוב שהזכרון נגמר נמוך יותר ולא ימחק את קבוצת התוים שלך. לפני הורדת תקרת הזכרון, עשה בדיקה קטנה של כמה זכרון נשאר על-ידי הכנסת:

```
PRINT FRE(0)-(FRE(0)<0)*64*1024
```

המכונה צריכה לענות עם 38908.

כעת הנמך את פסגת הזכרון באמצעות:

```
POKE 55,0:POKE 56,48
```

אחרי שדבר זה נעשה בדוק שנית את כמות הזכרון שנשאר ואתה צריך לקבל את הערך החדש 10237.

כעת בואו נחקור מעט את פעולת המקלדת, במיוחד את:

מחסנית המקלדת (KEYBOARD BUFFER)

כאשר מידע כלשהו נכנס אל המחשב דרך המקלדת, ה-64 שם אותו אוטומטית לתוך מחסנית המקלדת, שהיא עשרה בתי זכרון הממוקמים מ-631 עד 640. כדי לעקוב אחרי מה שבתוך המחסנית משמש בית נוסף אחר הממוקם ב-198. רק כדי לחקור את פעולת המחסנית, הכנס את תוכנית 3.1.

תוכנית 3.1

```
2 PRINT "<CLR>";
5 T$=CHR$(34)+CHR$(20)+"<RVSON>T<RVSOF>"
+CHR$(34)+CHR$(20)
6 M$=CHR$(34)+CHR$(20)+"<RVSON>M<RVSOF>"
+CHR$(34)+CHR$(20)
10 PRINT:PRINT"<HOME>";CHR$(34);:FOR X=0
TO 9
20 IF PEEK(X+631)=20 AND X<10 THEN PRINT
T$;:NEXT:GOTO 10
30 IF PEEK(X+631)=13 AND X<10 THEN PRINT
M$;:NEXT:GOTO 10
40 PRINT CHR$(PEEK(X+631));:NEXT:PRINT C
HR$(34):PRINT PEEK(198);:GOTO 10
```

כאשר אתה מריץ תוכנית זו, היא תדפיס בפינה השמאלית העליונה של המסך את תוכנה הנוכחי של מחסנית המקלדת. מתחת לזה היא תדפיס את מה שמאוחסן ב-198, כלומר את מספר התוים שקומודור 64 חושב שישנם במחסנית. מדוע מספר זה הוא אפס? ובכן, בשעת הדלקת המחשב ובזמנים אחרים מסוימים המחסנית מתמלאת ב"זבל" שלא צריך להיות שם. מה שכתובת 198 אומרת לך הוא הסיפור הרשמי. הקומודור 64 יקרא רק את מספר התוים שכתובת 198 מספרת לו שישנם שם, ויתעלם מן היתר.

כדי לראות את המחסנית בפעולה, הרץ את תוכנית 3.1 והכנס "FREDFREDFRED" ובזמן ריצת התוכנית תראה שהמחסנית תכיל כעת:
"FREDFREDFR"

והשורה מתחת אומרת לך כי 198 מכיל כעת את הערך 10.

כעת הכנס "SID" וראה מה מתרחש. "משהו מתרחש? לא, שום דבר לא מתרחש מפני שהמחסנית מלאה. זהו אמצעי של ראשון נכנס - ראשון יוצא, אשר משמעותו היא שהפריט הראשון שמוכנס - כלומר ה'F' של ה-FRED הראשון מוחזק מוכן להחזרה.

כדי לחקור עוד את מחסנית המקלדת, לחץ על RUN/STOP ושנה את תוכנית 3.1 כדי שתבצע את הדברים הבאים:

* תקבל את 10 הכניסות הראשונות מן המקלדת
* תבצע GET
* תאחסן את התוצאה ב-A$
* תציג את התו הראשון שיצא מן המחסנית

תוכנית 3.1(a)

```
45 IF PEEK(198)=10 THEN GETA$:PRINT"<4DC
RSR>"A$
50 GOTO 10
```

(הסר את GOTO 10 משורה 40)

כאשר אתה מריץ תוכנית זו חזור על התהליך הקודם על-ידי נסיון להכניס שלושה "FRED". כאשר תכניס את ה-E של ה-FRED השלישי, המחסנית תתמלא, כלומר 198 יכיל 10. אך מיד אחרי-כן, יתבצע GET ותו ילקח מן המחסנית ויאוחסן ב-A$ ויודפס על המסך. כך שהאות 'F' תלקח מן המחסנית ושאר 9 התוים יזוזו מיקום אחד לכיוון 9 המקומות הראשונים. כאשר מוכנסת האות הבאה 'E' היא תאוחסן ראשית במיקום 10 של המחסנית רק כדי לזוז הלאה כאשר 'R' מוצא על-ידי GET. אתה יכול כעת לעצור את התוכנית על-ידי RUN/STOP, אך אל תמחוק עדיין את התוכנית.

פניה למקלדת

לפני שנוכל למקם מחדש את קבוצת התוים, אנו צריכים לבצע דבר פורמלי אחרון, הכרחי בגלל התפקיד החשוב שממלאת המקלדת. מכיוון שהיא ערוץ הקשר העיקרי בין המשתמש ומערכת ההפעלה, המקלדת נסרקת ללא הרף. סריקה זו חייבת להיפסק או להכבות לפני העברת קבוצת התוים. הסריקה מופסקת על-ידי קְביעת סיבית אפס של 56334 לאפס

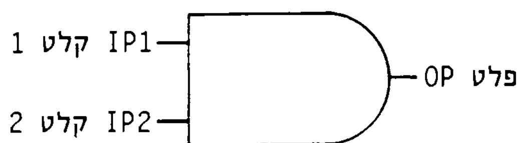
ומופעלת מחדש על-ידי קביעת סיבית זו לאחד. הבעיה היחידה בתהליך זה היא שרק סיבית אפס של 56334 צריכה להשתנות והשאר חייב להשאר כפי שהיה. כדי לעשות זאת ללא הרבה PEEK ו-POKE נשתמש ב:

פעולות לוגיות

פעולת AND – ביצוע AND לוגי

אולי יהיה ברור יותר אם נבדוק תחילה את השער האלקטרוני 'וגם' (AND) - הקומודור 64 שלך ממש מלא בהם! מה ששער 'וגם' עושה הוא קבלת שני אותות חשמליים, השוואה ביניהם והפקת פלט הבנוי על פי השוואה זו.

ציור 3.5 מראה אמצעי כזה עם קלט 1 וקלט 2 ופלט OP.



'AND' אלקטרוני

ציור 3.5

בפעולה, שער זה בוחן את IP1 ו-IP2 ואם שניהם קבועים ב-5 וולט, הרי ש-OP נקבע ל-5V. אך אם כל אחד מהם או שניהם קבועים לאפס וולטים, OP נקבע לאפס וולטים. במונחי מחשב, מצב 1 נחשב כ"אמת" והאפס כ"שקר", כך שחוקי שער 'וגם' הם: פלט השער הוא אמת אם קלט 1 וקלט 2 הם אמת. עבור כל המקרים האחרים הפלט הוא שקר או אפס. נוח לבטא מצבים אלו בטבלת אמת (TRUTH TABLE), ראה ציור 3.6.

IP1	IP2	OP
0	0	0
0	1	0
1	0	0
1	1	1

טבלת אמת עבור 'AND' אלקטרוני

ציור 3.6

כדי להשתמש בטבלה, קרא את השורה אשר בה נמצאים המצבים הדרושים של IP1 ו-IP2 וטור OP יתן את מצבו הלוגי של הפלט OP. למשל, אם נקח את ערך IP1=0 ו-IP2=1 שורה 2 תיתן את מצבו של OP כ-0 (שקר).

פעולת OR – ביצוע OR לוגי

כמו עם 'וגם', פעולה זו מבצעת השוואה לוגית של סיבית אחר סיבית בין שתי פיסות מידע. משמעותו של דבר זה הוא שכל סיבית מידע נבחנת ואם אחת מהן או השנייה שווה ל-1, הרי שמושג מצב אמת וסיבית התוצאה נקבעת ל-1.
דבר זה מובע בטבלת האמת, ציור 3.7. בטבלה זו שני הקלטים האפשריים נקראים IP1 ו-IP2 והפלט נקרא OP.

IP1	IP2	OP
0	0	0
1	0	1
0	1	1

טבלת אמת עבור 'או' לוגי
ציור 3.7

אופרטורים לוגיים ושמונה סיביות

עד כה, כל הפעולות הוצבו בפעולה על סיביות יחידות של מידע. אך כאשר פעולות לוגיות עוסקות במספרים בני שמונה סיביות, כל סיבית של מספר זה מטופלת לחוד. קח כדוגמא את המצב בו מבוצעת פעולת 'או' אל 100 (עשרוני)(01100100) ו-50 (עשרוני)(00110010). התהליך הוא פשוט פעולת 'או' של סיבית אחר סיבית. ציורים 3.8 ו-3.9 מראים שלב אחר שלב תהליך שבו ראשית האפס הראשון של 100 מושווה בפעולת 'או' עם האפס הראשון של 50 כדי לקבל אפס.
בהמשך, התו השני של ה-100 (אחד) מושווה עם התו השני של ה-50 (אפס) כדי לקבל אחד. ציור 3.9 מראה את השלבים השונים של התהליך יחד עם התוצאות.

כתוצאה מכך, כאשר 100 עובר פעולת 'או' עם 50 התוצאה היא:

	0	1	1	0	0	1	0	0	$= 100_{10}$
	0	0	1	1	0	0	1	0	$= 50_{10}$
=	0	1	1	1	0	1	1	0	$= 118_{10}$

ציור 3.8

אם נפענח זאת לפי טבלת האמת (ציור 3.7), ונתחיל מסיבית 0, נקבל:

סיבית 0: 0 בפעולת 'או' עם 0 נותן 0
סיבית 1: 1 בפעולת 'או' עם 0 נותן 1
סיבית 2: 1 בפעולת 'או' עם 1 נותן 1
סיבית 3: 0 בפעולת 'או' עם 1 נותן 1
סיבית 4: 0 בפעולת 'או' עם 0 נותן 0
סיבית 5: 1 בפעולת 'או' עם 0 נותן 1
סיבית 6: 0 בפעולת 'או' עם 1 נותן 1
סיבית 7: 0 בפעולת 'או' עם 0 נותן 0
כלומר 01110110

ציור 3.9

חזרה למשימה...

כמו שאמרנו, המקלדת היא אמצעי הקשר המרכזי בין המשתמש והמחשב ויש צורך לפעמים לשבור קשר זה.

כתובת זכרון אחת 56334 ($DC0E כאשר $ מסמל מספר הקסהדצימלי) מכיל סיבית מידע אחת - סיבית אפס - אשר מדליקה את המקלדת כאשר היא קבועה ל-1 ומכבה אותה כאשר היא מאופסת.

כדי לקבוע אותה נשתמש בשתי הפונקציות הלוגיות 'או' ו-'וגם'.

ראשית, כדי לקבוע רק סיבית אחת של הבית לאפס, נבצע AND בינה לבין בית המכיל רק אחדים חוץ מהמיקום לקביעה. וכך כאשר סיבית זו משמשת בפעולת AND יחד עם האפס היא נכבית אוטומטית לאפס.

לפיכך כדי לקבוע סיבית אפס לאפס, נבצע פעולת AND של הבית עם 254_{10} (11111110_2) כלומר:

מספר זה	101100111
בפעולת AND עם	111111110
נותן	101100110

ציור 3.10

אם אינך בטוח בתהליך, חסר כל מספר שברצונך מהמספר העליון ותראה כי המספר ישאר כשהיה מלבד סיבית אפס אשר תשתנה לאפס. שימוש בתהליך זה לכבוי המקלדת הוא:

תוכנית 3.2

```
X = PEEK(56334)
Y = X AND 254
POKE 56334,Y
```

נשים הכל בשורה אחת:

```
POKE 56334,PEEK(56334)AND 254
```

<u>אל תכניס</u> את זה כעת. אם תכניס לא תוכל להכניס את הפקודות להפעלת המקלדת שוב.

כדי להדליק שוב את המקלדת - סיבית אפס צריכה להקבע שוב ל-1. במקרה זה מבוצעת פעולת 'או' של הבית עם 1, כלומר שבע אפסים ו-1 בסיבית אפס. תהליך זה ישאיר את כל הסיביות אשר השתתפו בפעולת OR עם האפסים כפי שהיו אך יקבע את סיבית אפס ל-1. ציור 3.11 מראה תהליך זה בפעולה:

מספר זה	101100110
בפעולת 'או' עם	000000001
נותן	101100111

ציור 3.11

ומכאן, שתי השורות לכבוי ואחר-כך להדלקת המקלדת הן:

כבוי המקלדת	POKE 56334, PEEK(56334) AND 254
הדלקת המקלדת	POKE 56334, PEEK(56334) OR 1

גם תוכניות ההדגמה הקטנות שלנו (3.1 ו-(3.1(a) יכולות לעזור בתאור תהליך זה אם נכתוב אותן שיכבו את המקלדת, נניח אחרי ששלושה תוים הוכנסו. בעיה קלה עלולה להתעורר כאן, כי כאשר המקלדת מכובה לא נשארת דרך קלה לדבר עם המחשב. לכן פרוצדורת ההדלקה מחדש צריכה להיות מתוכנתת <u>לפני</u> נעילת המקלדת. שורות 60 ו-70 מבצעות את פעולת כבוי והדלקה עם עכוב מתאים באמצע.

תוכנית (3.2(a

```
47 IF PEEK(198)=3 THEN 60
60 POKE 56334,PEEK(56334)AND 254:PRINT"<
6DCRSR> KEYBOARD OFF":FOR X=1 TO 1000
70 NEXT:POKE 56334,PEEK(56334)OR 1:PRINT
"<UCRSR><4RCRSR> KEYBOARD ON":FOR X=1
TO 1000:NEXT
```

הרץ תוכנית זו וראה מה קורה כאשר אתה מכניס תוי מקלדת. אם רצונך בכך, שמור תוכנית זו. כאשר אתה מוכן להמשיך, הכנס NEW להסרת כל שורות התוכנית מן הזכרון.

שבב ה–VIC II וה–6510

המטרה הבאה לכבוש היא מיקומה מחדש של קבוצת התוים (CHARACTER SET), וכדי לבצע זאת יש לבדוק את תפקידו ל שבב ה-VicII.

פעולת שבבי ה-VIC נדונה כבר בפרק שנים, או בעצם חלק מפעולותיו הנוגעות בכתובות 53265, 53270 ו-53272. הקומודור 64 מכיל שבבים רבים ובתוכם שני אלו העוסקים בפעילות יום-יומית, ה-6510 ושבב ה-VicII. ה-6510 הוא המיקרופרוססור העיקרי בקומודור 64 ואילו שבב ה-VicII מנהל את כל פלט הווידאו אל המסך, ומשאיר את ה-6510 חופשי לעסוק במשימותיו שלו. פונקציות רבות דורשות את פעולתם ההדדית של אחד מהם עם השני ודבר זה נכון במיוחד בזמן תרגום פקודה למצב מסך אמיתי. כדי להבין כיצד זה עובד, צריך להביט על פעולת ה-ROM.

זוהי התוכנית הבנויה בתוך המכונה שלך ולמעשה גורמת למכונה לפעול. עליה להריץ את התוכנית במהירות רבה ולכן היא אינה כתובה בבייסיק אלא בשפה שהשבב עצמו משתמש - שפת מכונה.

בואו נקח משפט פשוט ונראה כיצד ה-6510 מטפל בו.

קח את המשפט:

PRINT "A"

דבר זה יגרום ל-64 להדפיס 'A' במיקום הסמן הבא.

הדבר הראשון שעל ה-6510 לעשות הוא להבין מה משמעות "PRINT". מכיון שזו מילת פקודה, היא אינה שמורה בזכרון כחמש אותיות נפרדות אלא כסמל (TOKEN), מספר בן בית אחד שהוא הקוד עבור PRINT. כך שה-6510 רואה 'PRINT', מחפש את הסמל הקשור אל הפקודה ואחר-כך מחפש ב-ROM סמל זה. כאשר הוא מוצא אותו, בטבלה, הוא מוצא שם גם כתובת, הכתובת היא כתובות ההתחלה של תוכנית שפת מוכנה אשר מבצעת 'PRINT'.

כעת הוא קופץ לכתובת זו ומתחיל להריץ את תוכנית שפת המוכנה המאוחסנת שם. המשימה הראשונה המבוצעת על-ידי תוכנית זו היא למצוא היכן להדפיס, כלומר לחפש בזכרון מהו מיקום הסמן העכשוורי. אחר-כך הוא בודק מה נמצא אחרי ה-'PRINT', וכאשר הוא מוצא סימן גרשיים (") הוא לוקח את ה-A ומשתמש באלגוריתם מיוחד כדי להפכו לקוד מסך אשר במקרה זה הוא 1. ה-6510 שם את ה-1 בכתובת המסך הנכונה. כעת הוא בודק מהו צבע התו העכשווי, ואחר-כך הוא מכניס את קוד הצבע המתאים לתוך אזור הצבע ב-RAM.

אחרי שגמר לבצע את כל המשימות הללו, יש לעדכן את מיקום הסמן, כלומר הזזתו מיקום אחד ימינה וכעת ה-6510 חופשי לחפש אחרי העבודה הבאה לביצוע ופשוט ממשיך בפעולתו.

בכל אופן, עדיין לא הודפס דבר על המסך, כל מה שה-6510 עשה היה לעדכן את אזור המסך ב-RAM. פה נכנס שבב ה-VicII!

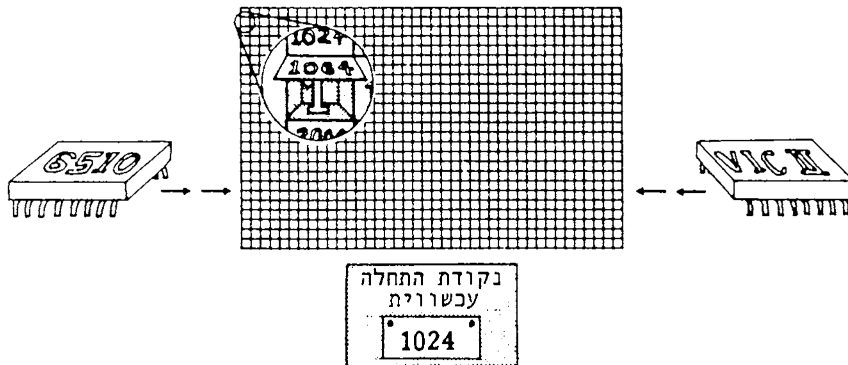
שבב ה-VicII מבצע משימות שליטת מסך רבות וללא הרף הוא סורק את זכרון המסך ב-RAM ומביט, בתורו, בתוכנה של כל כתובת זכרון המסך בנפרד, כאשר במקרה שלנו הוא מביט בכתובת 1064, הוא מוצא שם 1. זה חייב להתרגם לדמות המסך המתאימה עם צורה וצבע.

לפני שיוכל למצוא את מידע הצורה המתאים, ה-VicII חייב למצוא את הכתובת של מידע צורת התו. כדי לדעת זאת עליו להתיחס לשתי כתובות זכרון. הראשונה מאלו היא ב-56576 (למעשה רגיסטר מאחד משבבי CIA) והשניה היא ב-53272, שהיא אחת מהרגיסטרים של שבב

ה-VicII עצמו. אחרי שכתובות אלו נמצאו, הוא יכול להביט במידע השמור שם ולמצוא את תבנית הפיקסלים המתאימה. אחרי-כן הוא חייב למצוא את מידע הצבע המתאים לתו זה, ודבר זה הוא פשוט להסתכל באיזור הצבע ב-RAM אשר תמיד שמור באותו המקום.

אחרי שרכש את כל המידע הנחוץ, שבב ה-VicII יכול להמשיך הלאה ולהדפיס 'A' על המסך. ציור 3.12 ממחיש את כל התהליך.

שילוב VicII/6510 בזמן ביצוע - "PRINT"A



פעולות ה-6510	פעולות ה-VicII
* מציאת הסמל (TOKEN)	* בדיקת 1064
* מציאת השגרה	* מציאת מידע הפיקסלים עבור 1
* מציאת הסמן	* מציאת מידע הצבע עבור 1064
* מציאת הגרשיים (")	* הפקת תבנית פיקסלים
* חישוב קוד המסך עבור A	* הצגה על המסך
* אחסון הקוד עבור A	

ציור 3.12

באופן רגיל, כאשר ה-6510 מתבונן על הזכרון ב-53248 עד 57343 ($D000-$DFFF) הוא רואה את אזור הצבע (ב-$D800) ואת האזור הקרוי אמצעי קלט/פלט, כלומר שבב ה-VicII (ב-$D000), ה-SID (שבב השמע SOUND INTERFACE DEVICE) (ב-$D400) ושני שבבי CIA (COMPLEX INTERFACE ADAPTORS) (ב-$DC00 וב-$DP00). בכל אופן

אזור התוים ב-ROM נמצא גם הוא שם ועלינו לגרום ל-6510 לראות את אזור התוים ב-ROM במקום את איזור הצבע ב-RAM ואת אמצעי הקלט/פלט. דבר זה מושג בקלות על-ידי שנוי סיבית 2 של כתובת 1. זו קבועה באופן רגיל ל-1 אשר מאפשר ל-6510 לראות את אזור הצבע ואת אמצעי הקלט/פלט. אם סיבית 2 תשונה לאפס, ה-6510 יראה את אזור התוים ב-ROM במיקום זה.

לפני שנרוץ ונבצע זאת, אנו צריכים לחשוב מה יקרה לקומודור אם ה-6510 לא יוכל יותר לראות את זכרון הצבע ואת אמצעי הקלט/פלט. הבעיה האמיתית היחידה היא עם שבבי ה-CIA. שבבים אלו מטפלים בכל פעולות הקלט/פלט. מהמקלדת, אל הקסטה וממנה (אם קיימת) אל הדיסק וממנו (אם קיים) ל-RS232 (אם קיים). אם ה-6510 אינו יכול לראות את שבבי ה-CIA אי-אפשר לטפל בקלט או פלט אל אמצעים אלו. למראית עין, זו איננה נראית בעיה. ברור שאם נבטיח כי תוכנית CHAR.GEN לא עוסקת בקלט או פלט אל המקלדת, הטייפ, הדיסק או ה-RS232 בזמן שאנו מעתיקים את זכרון התוים ב-ROM אל ה-RAM, הרי שאין בעיה. או שיש? ובכן, כן. המקלדת היא הבעיה. ה-6510 בודק את המקלדת חמישים פעם בשניה לבדוק האם נלחץ אחד המקשים. כדי לבדוק את המקלדת זקוק ה-6510 למבט על שבבי ה-CIA אותו אין הוא מקבל אם נקבע את סיבית 2 של כתובת הזכרון 1 לאפס. הוא יראה את זכרון התוים וינהג בתו אותו ימצא כהוראה מן המקלדת - אסון!

סריקת המקלדת נמשכת כל הזמן באופן רגיל, התוכנית אשר הקומודור 64 מריץ כרגע מופרעת על-מנת לעשות זאת. אנו צריכים להפסיק הפרעה זו זמנית, בזמן שאנו משתמשים ב-6510 כדי להעתיק את זכרון התוים ב-ROM.

למזלנו דבר זה הוא קל למדי. מה שאנו עושים הוא להפסיק את שעון המערכת אשר מודד את חלקי חמישים של השניה ודבר זה עוצר את שגרת סריקת המקלדת מלהפריע (INTERUPT). כדי לעצור את השעון אנו צריכים לשנות את סיבית אפס של כתובת זכרון 56334 ($DCOE) לאפס. כדי להפעיל את השעון אחרי המהלך אנו משנים את אותה סיבית לאחד. מכיון שכתובת זכרון זו היא רגיסטר של אחד משבבי ה-CIA אנו יכולים לשנות את הערך רק כאשר ה-6510 יכול לראות את שבבי ה-CIA.

כך שתוכנית הפעולות תהיה:

1. עצירת שעון המערכת השולט על סריקת המקלדת על-ידי שנוי סיבית אפס של 56334 לאפס.
2. הענקת היכולת ל-6502 לראות את זכרון התוים ב-ROM במקום את שבבי ה-CIA על-ידי שנוי סיבית 2 של כתובת הזכרון 1 לאפס.
3. העתקת זכרון ה-ROM ל-RAM בעזרת ה-6502.
4. הענקת יכולת ל-6502 לראות שוב את שבבי הקלט/פלט על-ידי שנוי סיבית 2 של כתובת 1 חזרה לאחד.
5. הפעלתו מחדש של שעון המערכת על-ידי שנוי סיבית אפס של 56334 חזרה לאחד.

ואם נשים זאת בתוכנית:

תוכנית 3.2(b)

`60030 POKE 56334,PEEK(56334)AND 254`	עצירת שעון המערכת -
`60040 POKE 1,PEEK(1) AND 251: FOR I=0 TO 2047:POKE I+12288,PEEK(I+53248):NEXT`	6510 יראה ROM והעתקת התוים מ-ROM ל-RAM -
`60050 POKE 1,PEEK(1) OR 4` `60060 POKE 56334,PEEK(56334) OR 1`	הפעלתו מחדש של שעון המערכת -

לבסוף, צריך לומר לשבב ה-VicII היכן בדיוק אתה שם את קבוצת התוים החדשה. כדי למצוא היכן היא מאוחסנת, השבב מביט בכתובת 53272. כך שכדי לכוון מחדש את שבב ה-VicII לכתובת החדשה הכנס:

```
60070 POKE 53272,(PEEK(53272)AND 240)+12
```

נשתמש בשני תוים מיוחדים בתוכנית עצוב התוים ואלו יוגדרו מחדש בזמן ההתחלה. התוים שאנו נגדיר מחדש הם אלו המייצגים את 254 ו-255 ולכן הם מאוחסנים ב-:

12288 + (254*8) = 14320

12288 + (255*8) = 14328

התו שמיצג את 255 יהיה ריבוע מלא לחלוטין. כלומר הפיקסלים בכל שורה מסתכמים גם הם במקרה ב-255. ראה ציור 3.13.

1	1	1	1	1	1	1	1	= 255
1	1	1	1	1	1	1	1	= 255
1	1	1	1	1	1	1	1	= 255
1	1	1	1	1	1	1	1	= 255
1	1	1	1	1	1	1	1	= 255
1	1	1	1	1	1	1	1	= 255
1	1	1	1	1	1	1	1	= 255
1	1	1	1	1	1	1	1	= 255

ציור 3.13

ולפיכך כתובות 14328 עד 14335 ימולאו במספרי 255. כלומר:

```
FOR X = 0 TO 7: POKE(12288+255*8) + X,255: NEXT
```

ציור 3.14 מראה את התו השני, 254. לזה ישנם כמה פיקסלים ריקים המיוצגים על-ידי אפסים (0).

1	1	1	1	1	1	1	1	= 255
1	0	0	0	0	0	0	1	= 129
1	0	0	0	0	0	0	1	= 129
1	0	0	0	0	0	0	1	= 129
1	0	0	0	0	0	0	1	= 129
1	0	0	0	0	0	0	1	= 129
1	0	0	0	0	0	0	1	= 129
1	1	1	1	1	1	1	1	= 255

ציור 3.14

שני מספרים שונים צריכים להיות מאוחסנים כדי להגדיר תו זה. 255 לתוך כתובת אפס והכתובת השביעית, ו-129 לשש הכתובות האמצעיות, כלומר:

```
FOR X=1 TO 6:POKE(12288+254*8)+X,129
POKE(12288+254*8)+0,255
POKE(12288+254*8)+7,255
NEXT X
```

בזמן פעולתה, תוכנית יצירת התוים תתאים לעצובו מחדש של כל תו בקבוצת התוים, כלומר מ-0 עד 253 (תוכנית CHAR.GEN משתמשת ב-254 ו-255). שורות 60110 ו-60120 של תוכנית 3.3 מבקשות את ערכו של התו אשר ברצונך להגדירו ושומרות ערך זה במשתנה CH.

כאשר תריץ את התוכנית, אולי תתפלא על הזמן הארוך שלוקח להעביר את קבוצת התוים. בכל אופן, כאשר דבר זה נעשה אין צורך לעשותו שנית. למעשה, חשוב שדבר זה לא יעשה שנית. מכיון שאם תו הוגדר שנית - הרי שהעברת קבוצת התוים שנית תהרוס את ההגדרה החדשה.

דרך אחת לטפל בכך היא לשאול האם קבוצת התוים הועברה או לא, בעזרת INPUT או GET. אולם דרך שנונה יותר לעשות זאת קיימת, כאשר בזמן שגרת ההכנה, נקבע בית 53272 על-ידי הוספת 12 ובכך קביעת סיביות 3 ו-2 לאחד. וכך בדיקת בית זה תראה באם הוזזה קבוצת התוים או לא. מה שדרוש זה דרך להתבונן על הבית אך להתעלם מכל הסיביות חוץ מ-3 ו-2. דבר זה יכול להיות מושג על-ידי פעולת AND בין תוכנה של 53272 עם 12 ובדיקה האם ערך התוצאה הוא 12.

אם אינך בטוח בכך, בדוק זאת לפי הציור הבא:

	0 0 0 0 1 1 0 0
בפעולת AND עם 12	0 0 0 0 1 1 0 0
נותן	$0\ 0\ 0\ 0\ 1\ 1\ 0\ 0 = 12_{10}$
	1 0 1 1 1 1 0 0
בפעולת AND עם 12	0 0 0 0 1 1 0 0
נותן	$0\ 0\ 0\ 0\ 1\ 1\ 0\ 0 = 12_{10}$

ציור 3.15

ללא קשר מהו המספר שמופעל בפעולת AND עם 12, כל 1 חוץ מאלו שבסיביות 2 ו-3 נמחקים על-ידי אפסי ה-12. דבר זה ידוע לעיתים קרובות כמחיקת סיביות על-ידי פונקצית ה-AND.

ולכן דרושה שורת בייסיק אשר תאמר: אם תוצאת AND של המספר עם 12 שווה ל-12 אז דלג על שגרת העברת התוים. כלומר:

```
60010 IF PEEK(53272)AND 12) = 12 THEN 60080
```

צרוף כל החלקים השונים של פרוצדורת ההכנה נותן את השגרה כמו בתוכנית 3.3.

תוכנית 3.3

```
60000 POKE 53281,1
60010 IF (PEEK(53272)AND12)=12THEN60080

60020 PRINT"<CLR><BLU><11DCRSR><2RCRSR>
      PLEASE WAIT WHILE CHAR SET MOVED"
      :POKE55,0:POKE56,48
60030 POKE 56334,PEEK(56334) AND 254
60040 POKE 1,PEEK(1) AND 251:FOR I=0 TO
       2047:POKE I+12288,PEEK(I+53248):
      NEXT
60050 POKE 1,PEEK(1) OR 4
60060 POKE 56334,PEEK(56334) OR 1
60070 POKE 53272,(PEEK(53272) AND 240)+
      12
60080 FORX=0TO7:POKE 14328+X,255:NEXT
60090 FORX=1TO6:POKE 14320+X,129:NEXT
60100 POKE 14320,255:POKE 14327,255
60110 PRINT"<CLR><5DCRSR><BLU><4RCRSR>W
      HICH CHARACTER WOULD YOU LIKE"
60120 INPUT"<4RCRSR>TO DEFINE(0 TO 253)
      ";CH
```

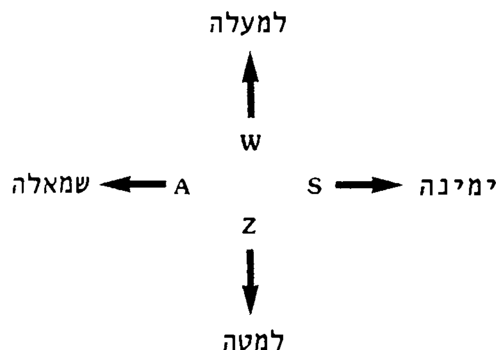
אחת מתכונות CHAR.GEN תהיה קלות הפעלתה ודבר זה תלוי עקרונית בגרפיקה. החלק העיקרי של הגרפיקה הוא מטריצת העצוב.

גירסה מוגדלת של מטריצת התו נקבעת באמצעות משפטי PRINT. אלו יוצרים מטריצת שמונה על שמונה (8x8) אשר בתוכה מעוצב התו החדש. שורות 60150 עד 60170 של תוכנית 3.4(a) מכילות חלק זה. בנוסף על הצגת התו כפי שעוצב בצורתו המוגדלת, התוכנית תאחסן למעשה את התו אל המסך ללא הגדלה. כל מה שדרוש לזה הוא POKE,CH:

תוכנית 3.4(a)

```
60140 PRINT"<CLR><4DCRSR><PUR>HERE IS Y
      OUR CHARACTER SO FAR:<GRN> ";:POK
      E 1214,CH
60150 PRINT"<HOME>"TAB(7)"<7DCRSR><RED>
      <RVSON>??????????":FOR X=1 TO 8
60160 PRINTTAB(7)"<RED><RVSON>?<GRN>>>>
      >>>>><RED>?"
60170 NEXT:PRINTTAB(7)"<RED><RVSON>????
      ??????"
```

כאשר התוכנית משמשת לעיצוב התו, הריבוע המלא נע בתוך המטריצה באמצעות ארבעה מקשים, כך:



ציור 3.16

כדי שהדבר יהיה ברור למשתמש, המקשים ותפקידיהם מוצגים על המסך, מסודרים באותה תבנית כמו על המקלדת. כדי לטפל בפעולתה המעשית של התוכנית, משמשים מקשי הפונקציות ושוב הם מוצגים על המסך באמצעות משפטי PRINT פשוטים. שורות 60180 עד 60240 בתוכנית 3.4(b) משלימות את המראה:

תוכנית 3.4(b)

```
60180 PRINT"<3DCRSR><GRN>       W-UP"
60190 PRINT" A-LEFT  S-RIGHT"
60200 PRINT"      Z-DOWN":PRINT"<RETURN>
      -END C.GEN";
60210 PRINT"<3UCRSR>"TAB(22)"<BLU>]<GRN
      >F1-SET PIXEL"
```

```
60220 PRINTTAB(22)"<BLU>]<GRN>F3-CLEAR
      PIXEL"
60230 PRINTTAB(22)"<BLU>]<GRN>F5-DISPLA
      Y DATA"
60240 PRINTTAB(22)"<BLU>]<GRN>F7-CLEAR
      BOARD<BLU>";
```

מקשי הפונקציות

ל-64 יש שמונה מקשי פונקציה: 1, 3, 5 ו-7 מתקבלים פשוט על-ידי לחיצה על מקשי הפונקציה לבדם; באמצעות SHIFT ומקשים אלו מופעלים מקשי הפונקציה 2, 4, 6 ו-8. כמו עם כל שאר המקשים על המקלדת לכל אחד יש את קוד ASCII אופייני השייך לו. וכך לחיצה על F1 (מקש פונקציה 1) יכולה להיות מזוהה על-ידי חיפוש אחר CHR$(133). CHR$(133) הוא קוד התו עבור מקש פונקציה אחד. הקודים הרלוונטיים עבור מקשי הפונקציה הם:

f1	CHR$(133)	f2	CHR$(137)
f3	CHR$(134)	f4	CHR$(138)
f5	CHR$(135)	f6	CHR$(139)
f7	CHR$(136)	f8	CHR$(140)

ציור 3.17

עיצוב התו

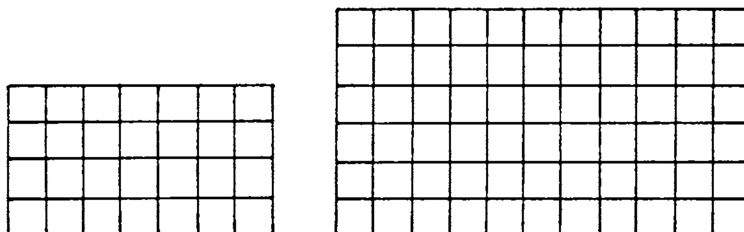
כדי לעצב כל תא נפרד צריך לעצב 64 פיקסלים, כלומר שמונה בתים של שמונה סיביות כל אחד, שבזמן העבוד אפשר לאחסן בזרם יחיד של 64 תוים. אך הבייסיק של ה-64 מספק דרך קלה בהרבה לעשות זאת בעזרת:

מערכים בשני מימדים

מערכים בשני מימדים משתמשים באותם שמות כמו מערכים במימד יחיד, ויכולים באותה צורה לשמש לאחסון מספרים ואינפורמצית תוים (STRING).

כלומר: A(x,y)
AB(x,y)
A9(x,y) וכו'...

למעשה, מערך דו-מימדי הוא מטריצה מלבנית של תאים אשר יכולים להיות מומחשים על-ידי שובך-יונים עם x שורות לכיוון אחד ו-y טורים לכיוון השני. ראה ציור 3.18.

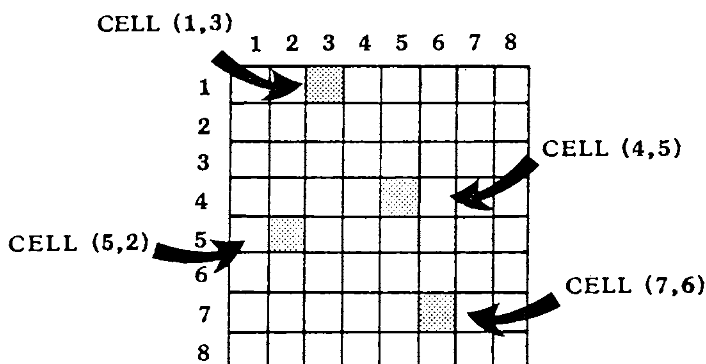


מערך 4*7
A(4,7)

מערך 6*11
W$(6,11)

ציור 3.18

במערך, אפשר לפנות לכל תא בנפרד על-ידי ציון הקואורדינטות שלו בכל כיוון. כלומר:



מערך של 8*8
ציור 3.19

במערך כמו בציור 3.19 אפשר להשתמש לאחסון התו בזמן הגדרתו כאשר כל שורה, כלומר A(1,1) עד A(1,8) מאוחסנת בבית אחד. כמו עם

מערכים בני מימד יחיד, צריך להגדיר ב-DIM את מגוון המערכים הדו-מימדיים

```
60000 POKE 53281,1:DIM A (8,8)
```

חזרה אוטומטית בכל המקשים

כדי להניע את הסמן סביב המסך משמשים מקשי השליטה בסמן שהם, לנוחות, מצוידים בתכונת חזרה אוטומטית. אך תוכנית CHAR.GEN משתמשת במקשי W, A, S ו-Z לשליטה בסמן ואלו אינם מצוידים באופן רגיל בחזרה אוטומטית. אך תכונה זו מופעלת בקלות באמצעות:

```
POKE 650,128
```

לפני תחילת עצוב התו, שגרת ההכנה נדרשת לקבוע את X ו-Y (הקואורדינטות של סמן העיצוב) ל-1 (שורה 60250). בנוסף, התו לעיצוב צריך להמחק מכיוון שלפני תהליך זה הוא מכיל אחד מהתוים הסטנדרטיים של ה-64.

```
60250 X=1:Y=1:POKE 55296 + 328,0
```

GET A$ משמש לקבלת קלט מן המקלדת ואמצעי זהירות שטוב לנקטו בזמן השימוש ב-GET A$ הוא להבטיח שמחסנית המקלדת לא תתמלא עם "זבל". אפשרי מאוד שהמשתמש ילחץ על שני מקשים או ילחץ על אחד מהם יותר מדי זמן, במיוחד אם הופעלה חזרה אוטומטית על כל המקשים. כפי שהוזכר בעבר, כתובת אחת בזכרון, 198, שומרת את מספר הבתים המאוחסנים במחסנית המקלדת. על-ידי קביעתה לאפס, הקומודור 64 משתכנע שהמקלדת ריקה. כך שכדי להריק את מחסנית המקלדת הכנס:

```
60280 POKE 198,0
```

התו המיצג את המיקום העכשווי של סמן העצוב משתנה לשחור והוא נע בתוך ריבוע העיצוב על-ידי הגדלת והקטנת קואורדינטות X ו-Y שלו. שורות 60300 עד 60330 בתוכנית 3.4(d) פשוט מחפשות את לחיצת מקשי השליטה ומניעות את המסמן לפיהם.

תוכנית 3.4(d)

```
60300 IFA$="A"ANDX>1THENX=X-1
60310 IFA$="S"ANDX<8THENX=X+1
60320 IFA$="W"ANDY>1THENY=Y-1
60330 IFA$="Z"ANDY<8THENY=Y+1
```

תפקידי מקשי הפונקציות

f1: קביעת פיקסל

אם נלחץ מקש F1 משמעותו אחסון מיקום הסמן העכשווי כאלמנט התו. אפשר לזהותו כאשר A$=CHR$(133) והמיקום במערך המתאים למיקום הסמן הנוכחי נקבע ל-1. כלומר:

```
60350 IF A$ = CHR$(133) THEN A(x,y) = 1
```

ואז הריבוע הנוכחי על רשת התו נקבעת לריבוע צבעוני מלא ונקבע בזכרון בצבע. המיקום במטריצת התו נמצא יחסית לפינה השמאלית עליונה במושגי x ו-y כלומר x+y*40 והוא מתוסף לקואורדינטה של נקודה זו - 1024+287. דבר זה נדרש כדי לשנות את המידע הנמצא בצורת מטריצה לצורה אותה יבין המסך, כלומר: בשורות של 40 תווים. חישוב דומה מחושב עבור זכרון הצבע. כאשר הדבר נעשה, השגרה נעה לשורה 60570 כדי לעדכן את התו הנוכחי המאוחסן בזכרון; כלומר:

```
60350 IF A$="<F1>" THEN A(X,Y)=1:
POKE 1024+287+X+Y*40,255:
POKE 55296+287+X+Y*40,6:GOTO 60570
```

f3: איפוס פיקסל

כאשר יש צורך לאפס פיקסל אשר נקבע בעבר, פונקציה זו עושה את המלאכה. היא פשוט הופכת את הפרוצדורה של F1 על-ידי קביעת המיקום הרלבנטי במערך לאפס, אחסון רווח במסך ולבסוף קביעת זכרון הצבע לצבע הרקע. בהיותה כה דומה לפונקצית קביעת הפיקסל הקוד כמעט זהה:

```
60360 IF A$="<F3>"THEN A(X,Y)=0:POKE
1024+287+X+Y*40,254:POKE 55296+287+X+Y*
40,5:GOTO 60570
```

f5: הצגת ה-DATA/איחסון בתוכנית

כאשר קוראים לפונקציה זו, התוכנית תשאל את המשתמש היכן לאחסן את משפטי ה-DATA עבור התו הנוכחי. אם יצויין מספר השורה אפס, יוצג DATA לתו זה על המסך אך לא יאוחסן. אם יצויין מספר שורה, תוכנס שורה לתוך התוכנית שתכיל את ה-DATA הדרוש לתו זה. זהו תהליך די מסובך ויוסבר בהמשך.

```
60370 IF A$ ="<F5>" THEN 60400
```

f7: ניקוי הלוח

ביסודה פונקציה זו מבטלת את עבודת העיצוב שנעשתה עד כה. במילים אחרות, היא מנקה את מטריצת העיצוב ומכינה אותה להתחלה חדשה. הפקודה CLR עושה זאת על-ידי קביעת כל המשתנים והמערכים חזרה לאפס, וברגע שזה נעשה התוכנית מנותבת חזרה אל שגרת הקלט. כלומר:

```
60380 IF A$ ="<F7>" THEN CLR:DIM A(8,8):GOTO 60110
```

f5: הצגת ה-DATA/איחסון בתוכנית - פרטים

שגרה זו מופעלת על-ידי לחיצת מקש פונקציה 5 ומבצעת שני דברים. השגרה מתחילה עם הודעה האומרת לנו מהם דברים אלו ומבקשת קלט מספר שורה, LN. בנסיון למנוע כתיבה על התוכנית אשר תשתמש בגרפיקה המעוצבת או על תוכנית CHAR.GEN עצמה, מומלצים מספרי שורות 50000 ו-10000. כדי לחזק "המלצה" זו נעשית בדיקת טעות בשורות 60400-60420 אשר מנתבת את התוכנית חזרה לקלט נוסף אם השורות הן מחוץ לטווח. כאשר הקלט התקבל, נמחקת ההודעה על-ידי הדפסת רווחים עליה, שורות 60430 ו-60440 בתוכנית 3.4(d). בנוסף על הקוד עבור F5, רשימה זו נותנת את התוכנית עבור שגרת עיצוב התו.

תוכנית 3.4(d)

```
60250 X=1:Y=1:POKE 55296+328,0
60260 FORB=0TO7:POKE 12288+B+CH*8,0:NEX
      T:POKE 650,255
60270 GETA$:IFA$=""THEN 60270
60280 POKE 198,0
60290 POKE 55296+287+X+Y*40,5+A(X,Y)
60300 IF A$="A" AND X>1 THEN X=X-1
60310 IF A$="S" AND X<8 THEN X=X+1
60320 IF A$="W" AND Y>1 THEN Y=Y-1
60330 IF A$="Z" AND Y<8 THEN Y=Y+1
60340 POKE 55296+287+X+Y*40,0:IF A$=CHR
      $(13) THEN 61000
60350 IFA$="<F1>"THENA(X,Y)=1:POKE1024+
      287+X+Y*40,255:POKE55296+287+X+Y*
      40,6:GOTO60570
```

```
60360 IFA$="<F3>"THENA(X,Y)=0:POKE1024+
      287+X+Y*40,254:POKE55296+287+X+Y*
      40,5:GOTO60570
60370 IF A$="<F5>" THEN 60400
60380 IF A$="<F7>" THEN CLR:DIM A(8,8):
      :GOTO 60110
60390 GOTO 60270
60400 PRINT"<HOME>ENTER LINE NO. FOR DA
      TA STATEMENT":PRINT"(10000-50000)
      :";
60410 PRINT"OR ZERO FOR DISPLAY ONLY":I
      NPUT LN
60420 IF (LN<10000 OR LN>49999 OR LN<>I
      NT(LN))AND LN<>0 THEN 60400
60430 PRINT"<HOME>";:FOR C=1 TO 3:FOR D
      =1 TO 38
60440 PRINT" ";:NEXT:PRINT:NEXT:PRINT"<
      HOME><18DCRSR>";
```

לפני הדפסת ה-DATA עצמו אל המסך, צריך למצוא את ערכי ה-POKE ודבר זה נעשה על-ידי קריאת הכתובות בהן הן שמורות. מרגע שהן נמצאו אפשר לאחסנם על המסך או בתוך מערך. בתוכנית זו אף אחת מדרכים אלו לא נבחרה. המידע מאוחסן כמשתנה אחד C$, הבנוי מהקריאות (PEEKS) המתאימות. כדי לעשות זאת, המחרוזת בנויה מערכי PEEK ופסיקים לחלופין. בעיה אחת נוצרת משיטה זו, נוכחותו של פסיק אחרון בשורה. זה מוסר על-ידי הוספת תו המחיקה (CHR$(20)) לסוף המחרוזת ומחיקת הפסיק האחרון. אם אינך בטוח בזאת, הכנס את תוכנית 3.5(a).

תוכנית 3.5(a)

```
10 C$="":FOR X=1 TO 10
20 C$=C$+"A"+","
30 NEXT X
50 PRINT C$
60 STOP
```

כאשר תוכנית זו מורצת, C$ צריך להיות מודפס כמחרוזת של עשרה תוי 'A' מופרדים על-ידי פסיקים עם פסיק בסוף. כעת שנה את התוכנית על-ידי הוספת שורה 40 של תוכנית 3.5(b).

תוכנית 3.5(b)

```
40 C$=C$+CHR$(20)
```

משהוספה השורה החדשה, המחרוזת C$ תכיל עשרה תוי 'A' המופרדים זה מזה על-ידי פסיקים אך הפסיק האחרון נמחק, כך שזה אכן עובד! כעת כשאתה מרוצה שהתוכנית עובדת, מחק את שורות 10 עד 60 על-ידי הכנסת מספרי השורות ולחיצת RETURN.

חזרה ל-CharGen

עד כה נוצרה שורת ה-DATA ואם מספר השורה יודפס על המסך לפניה, יווצרו כל מרכיבי משפט ה-DATA. עד כה משפט זה לא הוכנס לתוכנית וכעת יש לבצע זאת.

תוכניות המשנות את עצמן

במטרה להכניס שורה אל התוכנית, המחשב צריך להיות במצב עריכה, כלומר, עם סימן READY וסמן מהבהב. כמובן שהמחשב איננו במצב זה משום שהתוכנית עדיין רצה, התוכנית תהיה חייבת לפגוש 'END' כדי להפעיל את סימן ה-READY.
כאשר מבצעים עריכה ידנית יהיה התהליך הכנסת השורה ואז לחיצת מקש RETURN כדי להוסיף את השורה. עד כה הבנו כיצד להדפיס את השורה ואם נמשיך עם END, המחשב יחזור למצב עריכה. בחן זאת עם השורה הבאה. בזו ננסה לשנות תוכנית המדפיסה 'FRED' לאחת שמדפיסה 'FRED IS OK'. המטרה היא לשנות את התוכנית.

```
10 PRINT "FRED"
10 PRINT "FRED" -ל
20 PRINT "IS OK"
```

ולכן מה שנחוץ היא שורה שתיצור את שורה 20 ובעיה אחת שמתעוררת מיד היא זו של הדפסת גרשיים (") על המסך. למעשה זו אינה בעיה משום שקוד ה-ASCII 34 הוא סימן גרשיים.

```
PRINT CHR$(34)
```

צריך להדפיס סימן גרשיים על המסך. נסה זאת וראה. השלב הבא הוא להדפיס את כל השורה והפעם בוא ונעשה זאת מתוך התוכנית:

```
10 PRINT "FRED"
20 PRINT"20 PRINT";CHR$(34);"IS OK";C
HR$(34)
999 END
```

וכעת כשהיא תרוץ התוכנית תגיב ב-:

```
FRED
20 PRINT"IS OK"

READY
█ <CURSOR>
```

ציור 3.20

משלב זה, יצטרך הסמן לנוע מעלה ארבע שורות כדי שיהיה על שורה 20. דבר זה ניתן לעשות בתוכנית על-ידי הוספת שורה להעלאת הסמן מעלה ארבע שורות:

```
30 PRINT "<4UCRSR>"
```

כאשר זו נוספת והתוכנית מורצת המראה יהיה דומה לזה בציור 3.20, אך עם הסמן מהבהב על שורה 20. בשלב זה אם תכניס RETURN תוסיף את שורה 20 לתוכנית. לחץ עכשיו על RETURN ואחר-כך הכנס LIST 1-999 ותראה את התוכנית:

```
10 PRINT"FRED"
20 PRINT"IS OK"
30 PRINT"<4UCRSR>";
999 END
```

נאלצת ללחוץ בעצמך על מקש RETURN כדי להכניס את שורה 20 אל הזכרון. אנחנו רוצים למעשה שהמחשב יעשה זאת בעצמו. אנו יכולים לגרום למחשב לקרוא את מקש ה-RETURN ללא שנלחץ אנחנו RETURN על-ידי אחסון 1 בכתובת קריאת המקלדת (198), כלומר נלחץ איזה מקש, ואחסון ערכו של מקש ה-RETURN (13) ב-631 (מחסנית המקלדת). שנה את שורות 20 ו-30 כדי שיהיו:

```
20 PRINT"20 PRINT";CHR$(34);"IS OK";CHR$(34)
30 PRINT"<4UCRSR>";POKE 631,13:POKE 198,1
```

כעת הרץ את התוכנית והיא תגיב במראה הבא:

```
READY
20 PRINT"IS OK"
█ <CURSOR>
```

ציור 3.21

כעת הכנס LIST1-999 והתוכנית תהיה:

```
10 PRINT"FRED"
20 PRINT"IS OK"
30 PRINT"<4UCRSR>";POKE 631,13:POKE 198,1
999 END
```

עד כה מצוין. התוכנית משנה את עצמה. אך מה יקרה אם יוטל עליה להשתלב בתוכנית ארוכה יותר? כדי לבדוק זאת, שנה את התוכנית על-ידי הוספת השורות הבאות (התוכנית הולכת לעבור דרך לולאת PRINT):

```
20 PRINT"20 PRINT";CHR$(34);"IS OK";CHR$(34)
40 FOR X=1 TO 10
50 PRINT X
60 NEXT X
```

כעת הרץ את התוכנית. המסך יכוסה עם המספרים 1 עד 10. הכנס LIST1-999 ותראה כי שורה 20 לא שונתה כבעבר. הסיבה לכך שהמחשב לא יבדוק את מחסנית המקלדת עד שחישובי התוכנית יסתיימו (עד שתסתיים לולאת ה-FOR...NEXT), יש צורך לגרום למחשב להבחין ב-END או ב-INPUT כדי לאלף אותו לבדוק את מחסנית המקלדת. שנה את שורה 30 כדי שתהיה:

```
30 PRINT"<4UCRSR>";POKE 631,13:POKE 198,1:END
```

הרץ את התוכנית ואחר-כך הכנס LIST 1-999 ותראה כי שורה 20 שונתה שוב. אולם עלינו להבין כעת כי לולאת ה-FOR...NEXT לא בוצעה. נוכל להכריח את המחשב לבצע את הלולאה באמצעות GOTO 40. כל מה שדרוש הוא חזרה על התהליך שביצענו בשורה 20. כלומר:

* הדפסת שורה חדשה על המסך.
* העמדת הסמן על שורה זו.
* הכנסת RETURN למחסנית המקלדת.
* הבהרה למחשב כי ה-RETURN נמצא שם.
* אילוץ המחשב להתבונן במחסנית המקלדת כדי לראות מה לעשות הלאה באמצעות END.

כדי לשלב זאת בתוך התוכנית, ה-'GOTO' נוסף לתוכנית לפני שורת "מחסנית המקלדת" וזו משתנה להכיל CHR$(13) נוסף, כלומר:

תוכנית 3.6(a)

```
20 PRINT"20 PRINT";CHR$(34);"IS OK";CHR$(34)
25 PRINT"GOTO 40"
30 PRINT"<5UCRSR>";POKE 631,13:POKE 632,
13:POKE198,2:END
```

כעת סוף סוף, כאשר התוכנית מורצת היא תכניס את שורה 20 החדשה ואחריה 'GOTO', סמן מעליה ואחרי כן RETURN על שתי השורות, אחת עריכת שורה והשניה פקודת GOTO ישירה. עוד שיפור קטן נוסף היכול להתוסף לשורה זו הוא להדפיס את ה-'GOTO' בצבעי הרקע כך שהמשתמש לא יבחין בנוכחותו. דבר זה למעשה נעשה בשורה 60490 של תוכנית 3.6. אזהרה אחת - אם משהו בצבע אחר יהיה באזור זה ה-GOTO יראה! כעת אתה יכול למחוק את שורות 1-999 על-ידי הכנסת מספרי השורות והקשת RETURN.

כאשר ישולב תהליך זה לתוך תוכנית ה-CHAR.GEN הוא יכניס את שורת ה-DATA ואז יתחיל שוב את התוכנית. השגרה מוצגת בתוכנית 3.6(b).

תוכנית 3.6(b)

```
60450 IF LN>0 THEN PRINT LN;
60460 PRINT"DATA";:IF LN>0 THEN PRINT CH
;",";
60470 C$="":FOR C=0 TO 7:C$=C$+STR$(PEEK
(12288+
CH*8+C))+","
60480 NEXT:C$=C$+CHR$(20):PRINT C$:IF LN
=0 THEN
```

```
LN=1:GOTO 60500
60490 PRINT"<WHT>GOTO 60500<5UCRSR>":POK
E 631
,13:POKE 632,13:POKE 198,2:END
60500 PRINT"<HOME><2DCRSR><YEL><RVSON>PR
ESS
ANY KEY TO CONTINUE<RVSOFF><BLU>"
60510 POKE 198,0
60520 GET A$:IF A$="" THEN 60520
60530 PRINT"<HOME><2DCRSR><30 SPACES>"
60540 IF LN=0 THEN DIM A(8,8):GOTO 61000
60550 PRINT"<HOME><18DCRSR>";:FOR C=
1 TO 70:PRINT" ";:NEXT
60560 GOTO 60140
```

שגרה זו דורשת חלק מסוים אחד כדי ליצור תוכנית CHAR.GEN עובדת. זוהי השגרה אשר תקרא את מערך A(x,y), תתרגם את האינפורמציה לערכי POKE ותאחסנם בזכרון. שגרה זו תקרא בכל פעם שפיקסל חדש יתוסף לתו ולפיכך ערך ה-POKE לתו יתעדכן. מכיון שכל תא מן המערך, בכיוון x, מייצג ערך השווה לשתים בחזקת ערך מיקומו, נשתמש בלולאה לחישוב הערך המסכם של A, כלומר עבור תבנית הפיקסלים:

ערך המיקום -	7	6	5	4	3	2	1	0	
	0	1	1	0	0	1	1	0	
ערך סיבית -	0 +	2^6 +	2^5 +	0 +	0 +	2^2 +	2^1 +	0	
בעשרוני -	0 +	64 +	32 +	0 +	0 +	4 +	2 +	0	= 102

ציור 3.22

הסכום נעשה בעזרת הלולאה:

```
FOR B=1 TO 8:A=A(B,Y)*2↑(8-B):NEXT
```

אזהרה קלה: מספרי המיקום במערך מאוחסנים מימין לשמאל ואילו מספרי הסיביות, על-פי המסורת, משמאל לימין, כלומר:

ערכי x   1 2 3 4 5 6 7 8   מערך (x,y)

ערכי הסיביות 7 6 5 4 3 2 1 0

בדיקה עבור סיבית 3: A=A(B,y)*2↑(8-5)

כלומר A″¹≠²↑(3)

A=1*8=8

לאחר שחושב ערך הבית הוא מאוחסן בכתובת הנכונה והתוכנית חוזרת לקלט נוסף (שורה 60590 של תוכנית 3.6(c)), למעשה, כפי שנראה, השתמשנו ב-AND במקום לולאת FOR..NEXT

תוכנית 3.6(c)

```
60570 A=PEEK(12287+Y+CH*8):IF A(X,Y)=0
      THEN A=A AND(255-(2↑(8-X)))
60580 IF A(X,Y)=1 THEN A=A OR(2↑(8-X))
60590 POKE 12287+Y+CH*8,A:GOTO 60270
```

שגרת קריאת ה-DATA/יצירת התו

כאשר התוכנית משתמשת בתוים מוכנים שהוגדרו בעבר, צריך ליצור אותם ממשפטי ה-DATA ששם שמורה כל האינפורמציה הדרושה. בתוכנית CHAR.GEN התוים נוצרים בזמן הרצת התוכנית אך כאשר פונקציה זו משולבת עם שאר התוכנית, היא איננה מתאימה לעבודה הדדית עם תוכניות המשתמש עצמו. כדי להתגבר על בעיה זו, נוצרת שגרה קטנה אשר תקרא את ה-DATA ותאחסן אותו בכתובות הזכרון המתאימות.
לתוכנית 3.6(d) יש את היכולת ליצירת מספר מוגבר של תוים (N). ערכו של N יקבע על-ידי המשתמש כאשר יסיימו את התוכנית. עוד על כך בהמשך. קטע בתוכנית המוצע על-ידי תוכנית 3.6(d) ממוספר בנפרד מהגוף הראשי של תוכנית CHAR.GEN משום שיש בו צורך רק כאשר המשתמשים עיצבו את תויהם. שורה 10 מבטיחה כי התוכנית תקפוץ ישר אל התחלת תוכנית CHAR.GEN בשורה 60000.

תוכנית 3.6(d)

```
10 GOTO 60000

50000 FOR Y=1 TO N:READ A
50010 FOR X=0 TO 7:READ B
50020 POKE 12288+A*8+X,8
50030 NEXT:NEXT
50040 RETURN
```

מחיקת תוכניתך

כאשר סיימת את השימוש בתוכנית CHAR.GEN תהיה בכך עזרה רבה אם התוכנית תמחוק את עצמה, לא את כולה, כמובן. השורות שמכילות את מידע התוים שלך והשגרה לאחסון ערכים אלו בזכרון צריכות להשמד. לצערנו הקומודור 64 אינו מכיל פקודת מחיקה (DELETE) ולכן נדרשים כמה POKES מיוחדים.

שפת מכונה

חלק זה נפרד מתוכנית CHAR.GEN כך שמומלץ שתשמור (SAVE) את כל מה שיש בזכרון כרגע.
כפי שראינו בפרקים הקודמים, תפקודיו של ה-64 נשלטים על-ידי כתובות זכרון. ישנן כמה כתובות זכרון אשר שולטות בתוכנית ה-BASIC בזכרון. אלו הן כתובות 2049 והלאה. הסיום המדוייק של האיזור אותו שומר ה-64 למידע התוכנית אינו חשוב לנו, כל מה שאנו צריכים הוא כתובת ההתחלה, שהיא 2049.
ודא שאין שום תוכנית בזכרון על-ידי הכנסת NEW ולחיצת RETURN.
כעת הכנס פקודות ישירות אלו:

```
FOR X=2049 to 2060:PRINT X;PEEK(X):NEXT X
```

מה ששורה זו עושה הוא הדפסת התוכן המספרי של כתובות זכרון 2060-2049. אם רק כרגע הדלקת את המחשב או שאין שום תוכנית בזכרון הרי שהמצב צריך להיות בערך כך:

2049	0
2050	0
2051	88
2052	0
2053	140
2054	0
2055	112
2056	0
2057	0
2058	255
2059	255
2060	255

ציור 3.23

דבר זה הוא, עבור הקורא הממוצע של ספר זה, רק רשימת מספרים ואין לה כל מובן. כדי לתת לך מושג על האינפורמציה שמכילות כתובות אלו הכנס את שורה 1 ושוב את הפקודה הישירה:

```
1 PRINT A
FOR X=2049 TO 2060:PRINT ;PEEK(X):NEXT X
```

המצב ישתנה לכך:

	New	Old
2049	9	0
2050	8	0
2051	1	88
2052	0	0
2053	153	140
2054	32	0
2055	65	112
2056	0	0
2057	0	0
2058	0	255
2059	88	255
2060	0	255

כדי להקל על ההשוואה, הערכים הישנים של כתובות הזכרון ניתנים ביחד עם החדשים.

והנה משמעותן:

כתובות 2049 ו-2050 מכילות את כתובת הזכרון של השורה הבאה. הערכים הם הקסהדצימלים. הערך בכתובת 2050 הוא הבית היותר משמעותי והערך בכתובת 2049 הוא הפחות משמעותי. כך שהשורה הבאה של התוכנית תתחיל בכתובת:

$0809_{16} = 2057_{10}$

כתובת הזכרון 2057 היא כרגע אפס מכיוון שהכנסנו רק שורת תוכנית אחת. לפני שהכנסנו את שורה 1 היו הערכים של כתובות 2049 ו-2050 אפס. דבר זה אומר למחשב (ולנו) כי השורה הבאה אינה נמצאת בשום מקום משום שאין לנו שורה ראשונה. כדי להוכיח נקודה זו, הכנס את השורה הבאה:

```
POKE 2049,0:POKE 2050,0
```

כעת הכנס LIST, ואחרי לחיצת RETURN לא תראה את שורה 1.

אחסון אפס בכתובות אלו אומר למחשב כי אין שום תוכנית. כדי להדגים זאת בצורה טובה יותר, הכנס:

POKE 2049,9:POKE 2050,8

כעת משתכניס LIST תראה ששורה 1 חזרה. התוכנית לא נמחקה, המחשב פשוט שכח היכן היא התחילה. כאשר המחשב קורא את כתובות 2049 והלאה הוא מחפש שתי כתובות עוקבות המכילות אפס, דבר זה אומר למחשב היכן התוכנית מסתיימת. כעת אנו יודעים כיצד למחוק את התוכנית כולה, אך כמו שאמרנו קודם אנו רוצים למחוק רק חלקים מסוימים. כדי לעשות זאת עלינו ללמוד עוד על הדרך בה מאוחסנות שורות BASIC. קרא הלאה.

הכתובות הבאות שעלינו להתבונן בהן הן 2051 ו-2052. אלו מכילות את ערך מספר השורה. כתובת 2052 היא הבית המשמעותי יותר וכתובת 2051 מכילה את הבית הפחות משמעותי.

אם אין תוכנית בזכרון, הכתובות אשר משמשות למידע תוכנית ה-BASIC מכילות "זבל" וכך אפשר להתעלם מהערכים של קריאת כתובות 2049 עד 2060 כאשר אין תוכנית (כפי שנעשה קודם). הערכים בכתובות 2051 ו-2052 כאשר שורה 1 בזכרון הם:

0001

2051	2052
בית פחות	בית יותר
משמעותי (LSB)	משמעותי (MSB)

דבר זה מראה בברור כי מספר השורה הוא 1. כדי להדגים זאת יותר טוב הכנס:

POKE 2051,4
LIST

על-ידי אחסון 4 בכתובת 2051 שינינו את מספר השורה מאחד לארבע. הערכים הבאים הם הבעה מספרית של מה שנמצא בשורת התוכנית שלנו. בדוגמא שלנו המספרים הם:

קוד מספרי עבור PRINT	153	2053
קוד ASCII של רווח	32	2054
קוד ASCII של A	65	2055

לכל פקודת BASIC יש מספר מתאים אשר מאוחסן במקומה על-ידי המחשב. מספר הקוד עבור PRINT הוא 165. כמה מספרי קוד ופקודות ה-BASIC שלהם רשומים מטה:

מספר	פקודה	מספר	פקודה
155	LIST	205	FOR
162	NEW	207	DATA
167	THEN	209	INPUT
175	AND	213	GOTO
194	PEEK	214	RUN
200	LEFT$	217	GOSUB

על-ידי אחסון כל אחד מאלו, או ערכים אחרים, לתוך כתובת 2053, אנו יכולים לשנות את שורה 1 מ-PRINT A ל-

POKE 5053,175 = 1 AND A

POKE 2053,205 = 1 FOR A

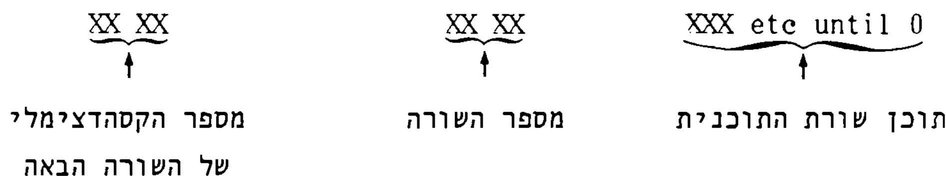
או POKE 2053,213 = 1 GOTO A

כתובות 2054 ו-2055 מכילות את המידע להמשך השורה. ואלו יכולות להשתנות באותה קלות כמו השורות האחרות:

משנה את הרווח ל-A POKE 2054,65

משנה את ה-A לרווח POKE 2055,32

כך שהחוק עבור כל שורת תוכנית השמורה בזכרון הוא כזה:



ידע זה יכול לשמש אותנו במחיקת חלק מן התוכנית.

הכנס את תוכנית הדוגמא:

```
10 REM THESE
20 REM LINES
30 REM ARE
40 REM STAYING
50 REM :@
60 REM ALL THESE
70 REM LINES ARE
80 REM GOING
90 REM GOODBYE
```

בתוכנית קצרה זו, שורות 60 והלאה ימחקו. שגרה קטנה תחפש בזכרון, מתחילה ב-2049 ומחפשת עבור שתי כתובות עוקבות שהראשונה מכילה ':' (קוד 58 ASCII) והשנייה מכילה "@" (קוד ASCII 64).
כאשר השגרה תמצא סימנים אלו היא תאחסן אפסים בכתובת המצביע המתאימה.

e pointer location is five memory locations behind the ':'

```
100 X=2049
110 IF PEEK(X)=58 AND PEEK(X+1)=64 THEN 130
120 X=X+1:GOTO 110
130 POKE X-5,0
140 POKE X-4,0
```

שגרה קצרה זו (שורות 100-140) מבצעת את הפעולה הדרושה. ודא שהכנסת את התוכנית נכון לפני שאתה מריץ אותה. כאשר היא מורצת תמחקנה שורה 50 והלאה ותשאר התוכנית הבאה:

```
10 REM THESE
20 REM LINES
30 REM ARE
40 REM STAYING
```

השורות הבלתי רצויות והתוכנית שמחקה אותן נעלמו.

תוכנית ה-CHAR.GEN נבנתה כך שכל השגרות שתמחקנה נמצאות בסוף התוכנית. הם גם יופרדו מקטע התוכנית שברצונינו לשמור על-ידי תו בדיקת מחיקה. כמו בדוגמא למעלה זה יהיה ":@".
שורה 60081 מכילה את מציין המחיקה ":@" ומפרידה את קטע התוכנית שברצונינו לשמור מזה שאנו עומדים למחוק. תוכנית 3.6(e) שואלת את המשתמש כמה תוים כבר עוצבו (N) ואם הם בטוחים שברצונם למחוק את התוכנית. במידה שכן שורות 62000 עד 62040 יוצרות שורות חדשות אשר התוכנית תהיה זקוקה להן כאשר שגרת המחיקה הופעלה.

```
60081 REM :@

61000 PRINT"<CLR><2DCRSR><5RCRSR>HAVE Y
      OU FINISHED USING"
61010 INPUT"<DCRSR><4RCRSR>CHAR.GEN (Y/
      N)";A$:IF A$="N" THEN 60110
61020 IF A$<>"Y" THEN 61000
61030 PRINT"<2DCRSR><4RCRSR>WARNING THI
      S ROUTINE DELETES THE"
61040 PRINT"<DCRSR><4RCRSR>CHARACTER GE
      NERATOR."
61050 PRINT"<2DCRSR><4RCRSR>ARE YOU REA
      LLY SURE THAT YOU'VE"
61060 INPUT"<DCRSR><4RCRSR>FINISHED (Y/
      N)";A$:IF A$="N" THEN 60110
61070 IF A$<>"Y" THEN 61060
61080 PRINT"<2DCRSR><4RCRSR>HOW MANY CH
      ARACTERS HAVE YOU":INPUT"<DCRSR><
      4RCRSR>REDESIGNED";N
61090 IF N<0 THEN 61080
62000 PRINT"<CLR><2DCRSR>"
62010 PRINT"10 PRINT";CHR$(34);"<CLR><B
      LU>";CHR$(34)
62020 PRINT"15 N=";N
62030 PRINT"20 GOSUB 60000:GOSUB 50000"
      :PRINT"60080 RETURN":PRINT"GOTO 6
      2050"
62040 PRINT"<HOME>": FOR X=631 TO 637:P
      OKE X,13:NEXT:POKE 198,6:END
62050 X=2049
62060 IF PEEK(X)=58 AND PEEK(X+1)=64 TH
      EN 62080
62070 X=X+1:PRINT"<HOME><13DCRSR><4RCRS
      R>";X:GOTO 62060
62080 POKE X-5,0
62090 POKE X-4,0
62100 END
```

חלק 3

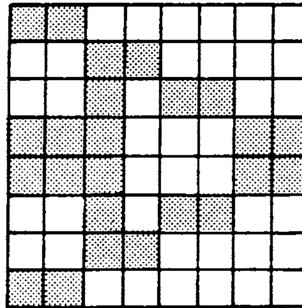
השימוש ב–Char.Gen

כדי לראות את CHAR.GEN בפעולה, בואו נעצב תו ונשלב אותו בתוכנית קצרה. ראשית נריץ את CHAR.GEN והמכונה תשאל:

WHICH CHARACTER WOULD YOU LIKE TO DEFINE?

הכנס 253

תו זה יוחלף על-ידי זה הנראה בציור 3.24.



ציור 3.24

כדי ליצור תו זה יש למלא את הריבועים המתאימים. הראשון שצריך למלאו הוא במקרה הריבוע הראשון בשורה. מכיון שהסמן נמצא כבר על ריבוע זה כאשר התוכנית מורצת, יש צורך ללחוץ F1 בלבד כדי לקבוע פיקסל זה. בשלב זה, ליד ההודעה

"HERE IS YOUR CHARACTER SO FAR"

תופיע הנקודה הראשונה של תו זה. כותרת התו יכולה להווצר על-ידי שימוש במקשים:

W הזזת הסמן מעלה

A הזזת הסמן שמאלה

S הזזת הסמן ימינה

Z הזזת הסמן מטה

וכדי לקבוע את התוים עצמם:

F1 - קובע את פיקסל הסמן הנוכחי כחלק מן התו.

F3 - מוחק את פיקסל הסמן הנוכחי.

F5 - מחשב את ערכי ה-POKE עבור התוים המעוצבים מחדש.

F7 - מנקה את התו שעוצב עד כה להתחלה חדשה.

הפעולות לציור שתי השורות הראשונות של התו הן:

- f1, S, f1, S
- Z, f1, S, f1, S
- Z, f1, S, f1, S
- Z, f1, S, f1
- Z, f1, A, f1, A
- Z, f1, A, f1, A
- Z, f1, A, f1, A
- Z, f1, A, f1
- W, W, S, S, f1
- W, f1, A, f1, A, f1
- W, f1, S, f1, S, f1
- W, f1, A, f1

עצור. זה אחד יותר מדי. כדי לתקן זאת לחץ על F3 כדי למחוק את הפיקסל האחרון. זכור, אם אתה לוחץ F1 במקום הלא נכון, F3 יתקן זאת. אם תמצא שיש צורך ביותר מדי שינויים, פשוט לחץ על F7 כדי לנקות את כל הלוח.

כעת משעצבת את כל התו ואתה בטוח שאין צורך בשינויים נוספים, לחץ על F5. המחשב ישאל:

ENTER LINE NO. FOR DATA STATEMENT

(10000-50000): OR ZERO FOR DISPLAY ONLY

רק כדי לבדוק את התוכנית הכנס אפס והמסך יראה את משפט ה-DATA המתאים מתחת למטריצת העצוב. בנוסף תנתן הודעה ל-:

PRESS ANY KEY TO CONTINUE

ברגע שמקש נלחץ, התוכנית תחזור למצב הכניסה שלה. כעת כדי להעביר את משפט ה-DATA לתוכניתך שלך, לחץ שנית על F5. תופיע הודעת 'AT WHICH LINE...' שנית ועכשיו עליך להכניס מספר שורה מתאים, נניח 10000. כאשר אתה לוחץ RETURN, משפט ה-DATA יופיע שנית ובעת עם מספר שורה. אחרי שניה בערך תופיע הודעת "PRESS ANY KEY" וכאשר תעשה זאת תחזור שוב אל הודעת "WHICH CHARACTER".

אם ברצונך לעצב תו נוסף עליך ראשית לנקות את מטריצת העצוב על-ידי לחיצת F7 והתחלה חדשה. תוכל להמשיך ולעצב את כל התוים שתצטרך ותמצא כמה שינויים מופיעים על המסך אם תעצב מחדש את התוים אשר עוצבו על-ידי CHAR.GEN עצמה! אך כרגע השאר עם תו אחד.

תוכנית 3.7 מדגימה טכניקה פשוטה לשימוש בשאריות התוכנית לעצוב תוים. התו המעוצב שלך מוצג על-ידי POKE אל המסך.

תוכנית 3.7

```
100 A=32
110 FOR X=1 TO 1000:POKE 1023+X,A:
A=PEEK 1024+X:POKE 1024+X,253:
POKE 55296+X,6:NEXT
120 END
```

האם אתה יכול לראות משורה 110 מה היא עושה? כל מה שהיא עושה זה אחסון התו לפני ספינת החלל ואז אחסונו לפניה כאשר היא נעה הלאה.

דבר אחד אחרון בקשר לעצוב תוים. אל תשכח שאתה משתמש בהם כאשר אתה משנה אותם.

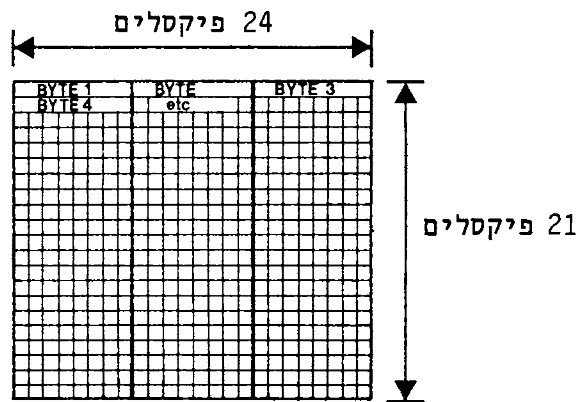
פרק 4
עולם הספרייטים

חלק 1 ספרייטים תוססים
סוגי הספרייטים

תכונה מתקדמת יותר של הקומודור 64 היא יכולתו להשתמש בספרייטים, או במונחי קומודור, עצמים ריבועיים נעים (MOVEABLE OBJECT BLOCKS - MOBs). אפשר להשתמש בבת-אחת בשמונה ספרייטים שונים, הממוספרים אפס עד שבע. אלו מתוארים טוב למדי על-ידי השם "עצמים ריבועיים נעים" מכיוון שהם בדיוק זה: ריבוע גדול, גודלו כ-7 תוים בערך, אשר נע בקלות על המסך. בנוסף קיימים אמצעים לבדיקת התנגשויות בין ספרייטים לספרייטים אחרים או לעצמים אחרים ברקע. בצורתו הפשוטה ביותר, הספרייט הוא תו גדול (3*2) בצבע יחיד. אפשר לעומת זאת להרחיבם על-ידי עצובם בשלושה צבעים או להגדילם לגודל כפול בכיוון האופקי, האנכי או בשניהם. וכך התכונות של הספרייטים הן:

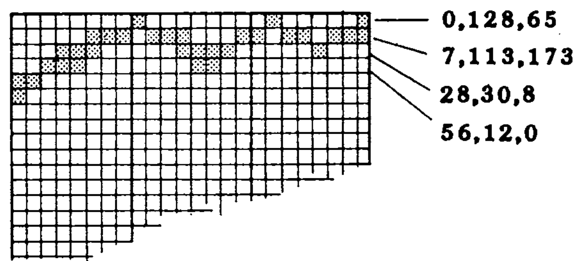
* גודל רגיל - 3 תוים על שני תוים
* גודל מורחב אופקית - 6 תוים על שני תוים
* גודל מורחב אנכית - 3 תוים על 5 תוים
* גודל מורחב בשני הצירים - 6 תוים על 5 תוים
* בשלושה צבעים, בכל אחד מן הגדלים למעלה (מצב רב צבעים)

למעשה גודל הספרייט מוגדר ביתר דיוק במושגי פיקסלים. כפי שציור 1 מראה, כל ספרייט בגודל רגיל תופס מלבן שרוחבו 24 פיקסלים בכיוון אופקי ו-21 בכיוון אנכי.



ציור 4.1

מכיון שספרייט בצבע יחיד בנוי מפיקסלים דלוקים או מכובים, הרי שלכל פיקסל דרושה סיבית אחת של מידע כדי להגדיר את מצבו. כדי לאחסן את כל המידע הדרוש להגדרת ספרייט, נדרשים כתוצאה מכך 24*21=504 סיביות זכרון. אלו מאורגנות בבתים, כפי שנראה בציור 4.1 כאשר שמונה הפיקסלים השמאליים למעלה שמורים בבית הראשון, שמונה הסיביות האמצעיות למעלה שמורות בבית 2 וכו'. החישובים למציאת ערך הבית לכל קבוצה של שמונה פיקסלים נעשית בדיוק באותה דרך כמו אלו של התוים הגרפיים בפרק 3. וכך הספרייט הנראה בציור 4.2 מוגדר על-ידי ערכי ה-POKE הבאים:



ציור 4.2

כדי לשלוט בספרייטים, נשמר אזור בזכרון המתחיל ב-53248. וכך הכתובות של פונקציות הספרייטים השונות ממוקמות ב-46 הבתים

העוקבים אחרי כתובות בסיס זו. מכיוון שקל הרבה יותר לזכור מספר בן שתי ספרות מאשר הכתובות בנות 5 הספרות, ניתנות כל כתובות פונקציות הספרייטים בצורת R+N, כאשר R = כתובת הבסיס של רגיסטר הספרייט ב-53248 ו-N = מספר הספרייט.

סוגי הספרייטים

הדיון עד כה עסק בפשוט ביותר מבין הספרייטים. אך קיימים אחרים, כפי שתואר למעלה. תכונותיו של כל ספרייט חוץ מהסוג בגודל הרגיל, בצבע אחד מתוארות למטה:

- **ספרייטים מורחבים אופקית**

  בית אחד, 53277 (R+29) שומר את האינפורמציה על גודלם האופקי של הספרייטים. כל סיבית מגדירה את גודלו של ספרייט אחד בלבד. לדוגמא, כדי להרחיב את ספרייט מספר 0 אופקית, נקבע את סיבית 0 של 53277 (R+29) ל-1. כאשר הספרייט מורחב אופקית, כל פיקסל אופקי מוכפל וכתוצאה מכך הרזולוציה נשארת 2 אף על פי שהספרייט תופס 48 פיקסלים על המסך.

- **ספרייטים מורחבים אנכית**

  בית אחד, 53271 (R+23) שומר את האינפורמציה על גודלם האנכי של הספרייטים. כמו בגודל האופקי, כל סיבית מגדירה את גודלו של ספרייט אחד בלבד. כאשר הספרייט מורחב אנכית, חוזרים על כל פיקסל אנכי של הספרייט וכך הרזולוציה נשארת 21.

- **ספרייטים מורחבים בשני הצירים**

  כאשר שתי הסיביות הרלוונטיות קבועות ל-1 מורחבים שני הצירים. וכך אם סיביות 2 של R+23 ושל R+29 קבועות ל-1 הרי שספרייט 2 מופיע כספרייט בגודל כפול.

- **חזרה על הגדרת גודל הספרייט**

  ציור 4.3 מראה את קביעת שני בתי הגודל וחוזר על ההשפעה שיש לכך על הספרייטים המתאימים.

רגיסטר	7 6 5 4 3 2 1 0	כתובת
R+23 אנכי	1 0 0 1 0 1 1 0	53271
R+29 אופקי	1 0 1 1 1 0 0 1	53277

מספר הספרייט	גודל אופקי	גודל אנכי
0	כפול	רגיל
1	רגיל	כפול
2	רגיל	כפול
3	כפול	רגיל
4	כפול	כפול
5	כפול	רגיל
6	רגיל	רגיל
7	כפול	כפול

ציור 4.3

ספרייטים רב-צבעוניים

כל שמונת הספרייטים יכולים להיות מוגדרים כרב-צבעוניים על-ידי קביעת הסיבית המתאימה ב-R+28 (53276) ל-1. כלומר:

	7 6 5 4 3 2 1 0	
R+28	0 1 1 1 0 0 0 1	53276

משמעותו היא שספרייטים 3,2,1 ו-7 הם בצבע יחיד וספרייטים 5,4,0 ו-6 הם רב-צבעוניים.

ספרייטים רב-צבעוניים יכולים להכיל עד שלושה צבעים שונים המוגדרים בדרך הרגילה של הקומודור 64. אך שניים מאלו חייבים להיות אחידים לכל הספרייטים ולפיכך כמה דוגמאות אפשריות אחדות הן:

ספרייט 0 אדום צהוב כחול

ספרייט 1 אדום צהוב לבן

ספרייט 3 אדום צהוב סגול

שיטת אחסנת מידע בצבע אחראית להגבלות אלו מכיוון שהניבלים (NIBBLE) הפחות משמעותיים (זכור - ניבל הינו חצי בית - כלומר 4 סיביות) של שני בתים מאחסנים את שני צבעי הספרייטים האחידים וניבל פחות משמעותי אחד עבור כל אחד מהצבעים המוגדרים בנפרד. כלומר:

מספר רגיסטר	השפעה
37	קובע צבע אחיד 1
38	קובע צבע אחיד 2
39	קובע את ספרייט 0 לצבע הספרייט
40	קובע את ספרייט 1 לצבע הספרייט
41	קובע את ספרייט 2 לצבע הספרייט
42	קובע את ספרייט 3 לצבע הספרייט
43	קובע את ספרייט 4 לצבע הספרייט
44	קובע את ספרייט 5 לצבע הספרייט
45	קובע את ספרייט 6 לצבע הספרייט
46	קובע את ספרייט 7 לצבע הספרייט

ציור 4.4

מכיון שספרייטים רב-צבעוניים מסובכים יותר מאלו יחידי הצבע, דרושה יותר אינפורמציה כדי להגדירם בשלמות. במצב רב-צבעוני כל פיקסל יכול להיות מכובה, בצבע 1, בצבע 2, או בצבע הספרייט. שתי סיביות מידע דרושות לאחסון ארבעת המצבים והן ממוקמות כדלהלן:

מצב סיבית 1	מצב סיבית 2	צבע התוצאה
0	0	רקע (שקוף)
0	1	צבע 1 (כללי)
1	1	צבע 2 (כללי)
1	0	צבע הספרייט (יחודי)

ציור 4.5

ולפיכך לספרייטים רב-צבעוניים יש רק מחצית הרזולוציה של ספרייטים יחידי צבע בכיוון האופקי (12 פיקסלים) מכיון שהסיביות הן בצמדים וכל צמד סיביות מכיל את האינפורמציה הדרושה להגדרת מרכיב אחד בן שני פיקסלים.

אם ננסה להשתמש בזאת, ציור 4.6 מראה את פינתו השמאלית העליונה של ספרייט, כפי שהוא מאוחסן וכפי שהוא נראה על המסך.

צמדי סיביות

00	01	11	11	11	10
00	01	11	11	11	
00	00	01	11	11	
00	00	01	11		
00	01	10	10		
00	00	01	10		

הספרייט כפי שהוא מאוחסן

B	C1	C2	C2	C2	CO
B	C1	C2	C2	C2	
B	B	C1	C2	C2	
B	B	C1	C2		
B	C1	CO	CO		
B	B	C1	CO		

הספרייט על המסך

B = צבע רקע

C1 = צבע 1

C2 = צבע 2

CO = צבע הספרייט

ציור 4.6

תוכנית יצירת ספרייטים: SPRITE.GEN

כמו בגרפיקה המעוצבת על-ידי המשתמש, יחקרו הספרייטים באמצעות פיתוחה של תוכנית ליצירת ספרייטים, תוכנית שרות אשר תקל על עצובו של הספרייט ועל שילובו בתוכנית המשתמשת בספרייטים. במבנה שלה, התוכנית דומה לתוכנית CHAR.GEN שפותחה בפרק 3, רק ההבדלים בין SPRITE.GEN ו-CHAR.GEN יכוסו בפרק זה.

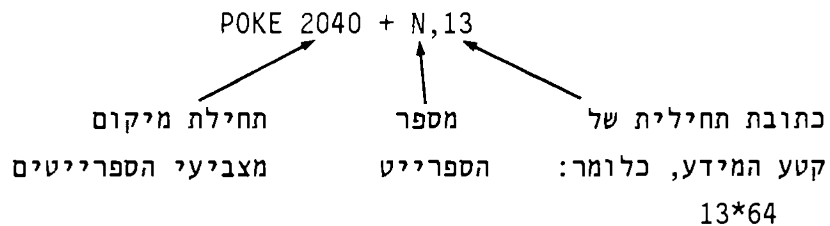
בזמן פעולת התוכנית, יבנה הספרייט פיקסל אחר פיקסל ויוצג ללא הרף על המסך. הספרייט שהכבוד נפל בחלקו הוא ספרייט אפס אך בסיומו של התהליך, העיצוב יכול לעבור לכל ספרייט אחר.

הכנה

אחרי שעוצבו הספרייטים, אפשר לאחסן את מידע הספרייטים בכל מקום בתוך קטע של 16K הנראה על-ידי שבב ה-VicII. במקרה שלנו משמעות הדבר 16K הראשונים של הזכרון. אך קיימת הגבלה אחת - הבית הראשון מבין ה-63 המרכיבים את הספרייט חייב להיות מאוחסן בתחילתו של קטע בן 64 בתים 0,64,128, וכו'... קטע אחד כזה נח בתוך מחסנית הקסטה הנמצאת בין 828 ל-1023. כתובת ההתחלה שלה היא 13*64 או 832, והפיקסלים של ספרייט ריק יכולים להיות ממוקמים על-ידי:

```
FOR X = 0 TO 62:POKE 832+X,0:NEXT X
```

כאשר עוצב הספרייט, יש לתת לו מספר וצריך להגיד לקומודור 64 כי ספרייט N ממוקם בקטע X. דבר זה נעשה בעזרת קטע בזכרון בין 2040 ו-2047 כך:



וכך כתובתו של ספרייט אפס מאוחסנת בכתובת 2040, וזו של ספרייט אחד ב-2041 וכו'.

וכך השורות הראשונות של SPRITE.GEN נראות:

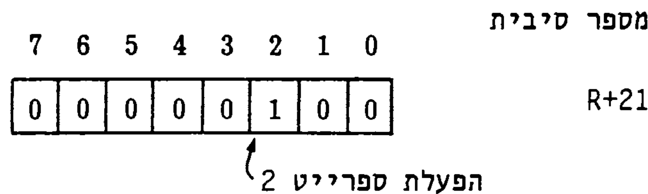
60000 PRINT "<CLR>" - נקוי המסך

60010 POKE 53280,3:POKE 53281,1: - קביעת צבע המסך
POKE 2040,13 - דיווח על ספרייט אפס בקטע 13

60020 FOR X=0 TO 62:POKE 832+X,0:NEXT - קבע את ספרייט אפס לספרייט ריק.

אחרי שעוצב הספרייט יש צורך להדליק אותו, או בשפת הקומודור להפעילו (ENABLE). דבר זה נעשה על-ידי קביעת סיביות נפרדות בבית R+21 לאחד. סיבית אפס שולטת בספרייט אפס, סיבית אחת בספרייט 1 וכו'.

כלומר:



POKE R+21,4

ולבסוף יש לתת לספרייט מיקום על המסך. קיימים 200 מקומות אפשריים בכיוון האנכי (Y) ו-320 בכיוון האופקי (X).
כדי להגדיר את המיקום האנכי (Y) על המסך, משתמשים בבית אחד עבור כל ספרייט, הבתים עצמם הם: R+1 עבור ספרייט 0, R+3 עבור ספרייט 1 ו-R+5 עבור ספרייט 2 וכו'. וכך כדי לקבוע את כיוון Y ל-100 עבור ספרייט 0, הפקודה הדרושה היא:

POKE R+1,100

במקרה של כיוון X החיים הם קצת יותר מסובכים מכיוון שצריך לפנות ל-320 נקודות אופקיות. מכיוון שחלק מן המספרים 0-319 הם גדולים מן המספר המקסימלי שאפשר לאחסן בבית יחיד, יש צורך למצוא יכולת אחסון חדשה. חשבון שמונה סיביות אינו מספק, כמובן, ולכן משתמשים בחשבון שש-עשרה סיביות. למעשה הדבר אינו מדויק - אמנם משתמשים בשני בתים אך למעשה דרושות רק 9 סיביות. הסיבית היחידה הנוספת הדרושה לכתובת X יכולה להמצא עבור כל שמונה

הספרייטים האפשריים בבית אחד, ומשתמשת ב-R+16 למטרה זו. וכמו בשיטות אחסון סיביות מתוחכמות אחרות, סיבית אפס משמשת כסיביות היותר משמעותית של כתובת X של ספרייט אפס. סיבית 1 עבור ספרייט 1 וכו'. בואו נחקור זאת על-ידי קביעת כתובת של 300 עבור ספרייט 3 ונחשב את ערכי ה-POKE הנדרשים.

הבית הפחות משמעותי אינו יכול לעבור את ה-255 ולכן סיבית אפס של הבית היותר משמעותי צריכה להקבע ל-1. מכיון שלכך יש ערך מקום של 256, צריך התוכן של הבית הפחות משמעותי להיות 44 (300-256). כלומר:

תוכנית 4.2

MSB									LSB								
0	0	0	0	0	0	0	1		0	0	1	0	1	1	0	0	
0+	0+	0+	0+	0+	0+	0+256		+	0+	0+32+	0+	8+	4+	0+	0		= 300

אם נשתמש בידע זה, השורות הבאות של SPRITE.GEN יכולות להכתב:

60030 POKE 53269,1: הפעלתו של ספרייט אפס

POKE 53249,74: קביעת ציר Y עבור הספרייט המוצג

POKE 53264,1:POKE 53248,35 קביעת ציר X עבור הספרייט המוצג

אחרי שנקבע הספרייט הבסיסי, צריך להגדיר את הפרמטרים האחרים, גודל וצבע. שגרה זו קובעת בפשטות את המשתנים ל:

* ספרייט מורחב אופקית - EX=1
* ספרייט מורחב אנכית - EY=1
* ספרייט רב-צבעוני - MC=1

צבע 1 - C1

צבע 2 - C2

צבע הספרייט - C0

שורות 60040 עד 60130 של תוכנית 4.3 מבצעות תפקיד זה וקובעות את הספרייט באמצעות פקודות POKE מתאימות.

```
60050 PRINT"<GRY1><CLR><4DCRSR>"TAB(12)
      ;:INPUT"ENLARGED X    <3LCRSR>";A$
      :IFA$="Y"THENEX=1:GOTO60060
60055 IFA$<>"N"THEN60050
60060 PRINTTAB(12);:INPUT"<3DCRSR>ENLAR
      GED Y    <3LCRSR>";A$:IFA$="Y"THEN
      EY=1:GOTO60070
60065 IFA$<>"N"THENPRINT"<5UCRSR>":GOTO
      60060
60070 MC=0:PRINTTAB(11);:INPUT"<3DCRSR>
      <RED>M<BLU>U<PUR>L<ORNG>T<BRWN>I<
      GRY1>C<GRN>O<LTRED>L<GRY2>O<LTBLU
      >U<RED>R<GRY1>    <3LCRSR>";A$:IFA
      $="N"THEN60120
60075 IFA$<>"Y"THENPRINT"<5UCRSR>":GOTO
      60070
60080 MC=1
60090 INPUT"<RED><2DCRSR><3RCRSR>MULTIC
      OLOUR #1 (.<RVSON> <RVSOF>)<GRY1>
          <5LCRSR>";C1
60095 IF C1<0ORC1>15THENPRINT"<4UCRSR>"
      :GOTO60090
60100 INPUT"<BRWN><DCRSR><3RCRSR>MULTIC
      OLOUR #2 (<RVSON>  <RVSOF>)<GRY1>
          <5LCRSR>";C2
60105 IF C2<0ORC2>15THENPRINT"<3UCRSR>"
      :GOTO60100
60110 POKE53285,C1:POKE53286,C2
60120 IFMC=0THENPRINTTAB(4);
60122 PRINT"<BLU><2DCRSR><4RCRSR>SPRITE
       COLOUR ";:IFMC=1THENPRINT"(<RVSO
      N> <RVSOF>.)";
60124 INPUT"<GRY1>      <5LCRSR>";CO$:CO
      =VAL(CO$):IFCO=0ANDCO$<>"0"THENCO
      =99
60126 IF CO<0ORCO>15THENPRINT"<4UCRSR>"
      :GOTO60120
60130 POKE53287,CO:POKE53277,EX:POKE532
      71,EY:POKE53276,MC
```

מראה המסך

בגלל המסך הגדול יותר הנדרש על-ידי הספרייטים, לא נשאר מקום לסמן שהיה בתוכנית CHAR.GEN. כמו כן, הרשת הראשונית חייבת להיות רשת של נקודות. כתוצאה מכך מתקבלת תוכנית פשוטה יותר, כפי שיעידו שורות 60140 עד 60170.

תוכנית 4.4

```
60140 PRINT"<CLR><3DCRSR>";TAB(27);"<BL
      U>YOUR":PRINTTAB(27);"SPRITE"
60150 PRINT"<HOME><DCRSR><RED><RVSON>
                                ":FORX=1T
      O21
60160 PRINT"<RED><RVSON> <RVSOF><GRN>..
      ......................<RED><RVSON
      > <RVSOF>"
60170 NEXT:PRINT"<RED><RVSON>
                          <BLU>"
```

קביעת הספרייט והמצב

כמו ב-CHAR.GEN, מגודר מערך עבור אחסון המידע (DIM), והפעם הוא גדול יותר: 24*21. מיקומו הראשוני של הסמן נקבע ל-1 (X=1,Y=1). וכאשר דבר זה נעשה, הספרייט נקבע לאפס בזכרון ומופעלת חזרה אוטומטית של כל המקשים.

כאשר נוצר קלט, נקבע מחדש צבע הסמן העכשווי ואז מפורש הקלט בשורות 60300 עד 60340 בתוכנית 4.5. צבע הסמן נקבע חזרה לאפס (שחור) ונעשית בדיקה עבור קלט של RETURN (CHR$(13)). מוצגת צורתו העכשווית של הספרייט בימין המסך על-ידי שורת 60400 עד 60450.

תוכנית 4.5

```
60250 X=1:Y=1:POKE55296+81,0:IFZX=0THEN
      ZX=1:DIMA(24,21)
60260 FORB=0TO62:POKE832+B,0:NEXT:POKE6
      50,128:POKE53269,1
60270 GETA$:IFA$=""THEN60270
60280 POKE198,0
60290 POKE55296+40+X+Y*40,5+A(X,Y)
60300 IFA$="A"ANDX>1THENX=X-1
```

```
60310 IFA$="S"ANDX<24THENX=X+1
60320 IFA$="W"ANDY>1THENY=Y-1
60330 IFA$="Z"ANDY<21THENY=Y+1
60340 POKE55296+40+X+Y*40,0:IFA$=CHR$(1
      3)THEN61000
```

כעת נערכות בדיקות עבור לחיצה על מקשי הפונקציה, בדיוק כפי שנעשה ב-CHAR.GEN.

f1: קבע את הפיקסל הנוכחי בספרייט

פונקציה זו היא בעצם זהה לפונקציה המתאימה ב-CHAR.GEN. היא קובעת קודם כל את התא המתאים במערך A(X,Y) ל-1, מדליקה את כתובת המסך וקובעת את צבע המסך. אחרי-כן היא שולחת את התוכנית לשגרה ב-60570 שם הפיקסל נקבע בזכרון. כלומר:

תוכנית 4.6

60350 IF A$="<f1>"THEN A(X,Y)=1:	קביעת מערך
POKE 1024+40+X+Y*40,160:	קביעת פיקסל במסך
POKE 55296+40+X+Y*40,6:	קביעת צבע המסך
GOTO 60570	GOTO לעדכון הספרייט

המטרה הראשונה היא חשוב מספר הבית (C) למיקום מערך נתון X. דבר זה מוגדר על-ידי C=INT((X-1)/8).

אחרי כן, יש צורך לחשב את ערכי ה-POKE עבור הבתים הנפרדים באמצעות לולאה המסכמת את ערכי הסיביות הנפרדות.

תוכנית 4.7

60570 C=INT((X-1)/8)	חשוב מיקום הבית
60575 A=PEEK(829+Y*3+C):IFA(X,Y)=0THENA=AAND(255-(2↑(8-(X-(C*8))))) 60580 IFA(X,Y)=1THENA=AOR(2↑(8-(X-(C*8))))	סכום ערכי סיביות
60590 POKE829+Y*3+C,A:GOTO60270	אחסון הערך בזכרון

f3: כבה את הפיקסל הנוכחי בספרייט

פונקציה זו דומה מאוד לפונקציה F1 אך בהיפוך. מעט הסבר נראה הכרחי.

תוכנית 4.8

60360 IF A$="<f3>"THEN	אפס מערך A(X,)=0
POKE 1024+40+X+Y*40,46:	אפס את הפיקסל על המסך
POKE 55296+40+X+Y*40,5:	קבע את צבע המסך לצבע הרקע
GOTO 60570	GOTO לעדכון הספרייט

f5: הדפס את ה-DATA על המסך

מכיון שהספרייטים גדולים הרבה יותר מתוים נפרדים, אין אפשרות להציג על המסך את מידע הספרייט ואת מטריצת העצוב באותו הזמן. יש צורך, איפוא, להפריד את שני התהליכים כך ש-F5 פשוט יכסה את שגרת "הצג על המסך".

כדי להכין את המסך, ההודעה "YOUR" ו-"SPRITE" נמחקות והספרייט מכובה (POKE 53296,0). אחר מתקבלים שלושת הבתים הראשונים (FOR D=0 TO 2) על-ידי PEEK לתוך כתובת האחסון ואלו מוכנים להדפסה אל המסך. שורה 60410 עושה זאת על-ידי קיצוץ המקום המתוסף על-ידי ה-BASIC של ה-64 ואחר מדפיסה זאת ואת הפסיק הנוסף. אחרי שהלולאה 0 עד 2 מתבצעת, מוגדלת הלולאה של 0 עד 20 כדי לקרוא את שלושת הבתים הבאים של הספרייט. כאשר המשתמש סיים עם ה-DATA ולחץ על מקש אחר, המסך חוזר להיות כפי שהיה קודם.

תוכנית 4.9

```
60370 IFA$="<F5>"THEN60400
```

f6: כתוב את ה-DATA אל התוכנית

מכיון שדבר זה זהה לתוכנית CHAR.GEN אין צורך בהסברים.

תוכנית 4.10

```
60375 IFA$="<F6>"THEN60460
```

f7: נקה את מטריצת העיצוב

כדי לבצע את נקוי המטריצה, יש צורך רק לאפס את המשתנים השונים ואת המסך. הפקודה CLR עושה את העבודה הראשונה ו-GOTO 60000 את השניה. שורה 60390 משלימה את לולאת הקלט על-ידי קפיצה חזרה ל-GET.

תוכנית 4.11

```
60380 IF A$ = "<F7>" THEN CLR:GOTO 60000
60390 GOTO 60270
```

משפטי איחסון הספרייט ב-DATA

כאשר מכובה ספרייט העבודה (POKE 53269,0), בא התחקיר היכן מאוחסן הספרייט, באיזה הבדלי שורה, ולבסוף יש בדיקת-טעות בנסיון למנוע מחיקה (שורה 60475).

תוכנית 4.12

```
60460 POKE53269,0
60465 PRINT"<RED><CLR><3DCRSR>AT WHICH
      LINE DO YOU WISH TO STORE YOUR"
60470 INPUT"   SPRITE DATA(10000-60000)
      ";LN
60475 IFLN<10000ORLN>59999THEN60465
60480 INPUT"<3DCRSR>LINE NUMBER INCREME
      NTS OF";SP
```

אחרי כן מוכנסים מספר הספרייט וקטע האחסון שלו.

תוכנית 4.13

```
60485 INPUT"<3DCRSR>WHICH SPRITE WILL T
      HIS BE(0-7)    <4LCRSR>";SN
60487 IFSN<0ORSN>7THENPRINT"<5UCRSR>":G
      OTO60485
60490 PRINT"<3DCRSR>AT WHICH BLOCK DO Y
      OU WANT THE DATA TO"
60492 INPUT"BE STORED (0-255)      <5LCR
      SR>";BN
60494 IF BN<0ORBN>255THENPRINT"<6UCRSR>
      ":GOTO60490
```

אחרי שנאסף המידע הכללי של הספרייט, יש לאחסנו לפני העברתו למשפטי DATA. מכיון שיתר ה-DATA מוכנס לזכרון באמצעות POKE, הגיוני לאחסן את המידע הכללי על הספרייט באותה דרך. תהליך זה נעזר עוד בהמצאותם של מספיק בתים ריקים מתחת למחסנית הקסטה לאחסון המידע הכללי של הספרייט. שורות 60495 ו-60500 שומרות את שמונה הבתים הללו.

תוכנית 4.14

```
60495 POKE824,SN:POKE825,BN:POKE826,CO:
      POKE827,EX
60500 POKE828,EY:POKE829,MC:POKE830,C1:
      POKE831,C2
```

לאחר שבתים אלו נמצאים במקומם, תאחסן שגרה המועלית פעם אחת את המידע הכללי והייחודי במשפטי DATA. אחת מתכונותיה החשובות של שגרה כזו היא שה-DATA מאורגן לפי סדר. כאשר דבר זה מבוצע, אפשר להשתמש ב-DATA לבניה מחדש של ספרייטים לפי הצורך. השימוש ב-DATA והכנסתו לתוכנית עובדים באותה הצורה ששמשה לשגרת ה-DATA בלבד.

תוכנית 4.15

```
60505 PRINT"<BLU><CLR>":FORX=0TO6:PRINT
      :PRINTLN+X*SP;"DATA";:FORY=0TO9
60510 A$=STR$(PEEK(824+X*10+Y)):PRINTRI
      GHT$(A$,LEN(A$)-1);:PRINT",";
60515 NEXT:PRINTCHR$(20);:NEXT:A$=STR$(
      PEEK(824+X*10+Y))
60520 PRINT","RIGHT$(A$,LEN(A$)+1):PRIN
      T"<WHT>GOTO60530<HOME>";
60525 FORX=0TO7:POKE631+X,13:NEXT:POKE1
      98,8:END
60530 PRINTTAB(6)"<RED><2DCRSR><RVSON>
      PRESS ANY KEY--TO CONTINUE <RVSOF
      >";
60535 GETA$:IFA$=""THEN60535
60540 GOTO61000
```

אחרי ש-SPRITE.GEN שמשה ליצירת ספרייט ואחסונו בזכרון, התוכנית שתשתמש בו תצטרך להיות מסוגלת לפענח זאת בצורה משביעת רצון. ההשלכות של דבר זה הן שהסדר והכתובת של האחסון חייבים להיות מוגדרים בצורה כזו שהשגרה הקוראת תבצע את מטרתה בסדר הנכון.

זוהי השגרה אשר תשאר אחרי ששגרת החיסול תתבצע. השלבים שהיא עוברת הם:

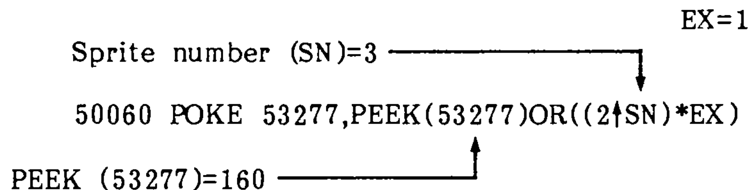
* קביעת כל הספרייטים לגודל רגיל וצבע אחיד.
* קריאת המידע הרגיל של הספרייט כדי להגדיר קטע, מספר, גודל, צבע וכו'.
* קריאת מידע הפיקסלים ואחסונו במקום הנכון. הקצאת המידע הכללי של הספרייט.

רק השלב הלפני אחרון מעורר בעיות מסוימות, מכיון שבתים יחידים משמשים לאחסון מידע עבור כל שמונה ספרייטים. אם נקח את שורה 50060 כדוגמא, זו שומרת את מידע הרחבת X. נניח כי ספרייט 3 צריך להיות מורחב בכיוון X וספרייטים 5 ו-7 כבר נקבעו למורחבים. במקרה זה EX נקבע ל-1. ואם נעבור דרך שורה 50060: ספרייטים 5 ו-7 כבר קבועים כמורחבים וכתוצאה מכך כתובת 53277 נראית כ:

7	6	5	4	3	2	1	0
1	0	1	0	0	0	0	0

128+ 0 +32 + 0 + 0 + 0 + 0 + 0 = 160

כלומר PEEK(53277) יתן 160.



והשורה הפשוטה יותר היא:

50060 POKE 53277,(160 OR 2↑3*1)

כלומר 50060 POKE 53277,(160 OR 8*1)

כלומר 50060 POKE 53277,(160 OR 8)

כלומר

1	0	1	0	0	0	0	0

0	0	0	0	0	1	0	0

1	0	1	0	0	1	0	0

128 +0 +32 +0 +8 +0 +0 = 168

ושורה 50060 הופכת:

50060 POKE 53277,168

תהליך דומה משמש לקביעת הרחבת Y ומצבים רב-צבעוניים בשורות 50070 ו-50080.

תוכנית 4.16

```
50000 POKE53277,0:POKE53271,0:POKE53276
      ,0
50010 FORY=1TON:READSN,BN,CO,EX,EY,MC,C
      1,C2:FORX=0TO62
50020 READC:POKE(BN*64)+X,C
50030 NEXT
50040 POKE2040+SN,BN
50050 POKE53287+SN,CO
50060 POKE53277,PEEK(53277)OR((2↑SN)*EX
      )
50070 POKE53271,PEEK(53271)OR((2↑SN)*EY
      )
50080 POKE53276,PEEK(53276)OR((2↑SN)*MC
      )
50090 IFMC=1THENPOKE53285,C1:POKE53286,
      C2
50100 NEXT:RETURN
59999 REM :@
```

החלק האחרון של תוכנית SPRITE.GEN הוא זה שמוחק את החלק הלא רצוי של התוכנית. שגרת מחיקה זו היא כמעט זהה לשגרת המחיקה של CHAR.GEN, ההבדל היחיד הוא ששגרת המחיקה של הספרייטים מסירה את כל השורות מ-60000 והלאה.

```
61000 PRINT"<BLU><CLR><2DCRSR><4RCRSR>H
      AVE YOU FINISHED USING"
61010 INPUT"<DCRSR><4RCRSR>SPRITE.GEN (
      Y/N)";A$:IFA$="N"THEN60050
61020 IFA$<>"Y"THEN61000
61030 PRINTTAB(4)"<2DCRSR>DO YOU WISH T
      O CREATE THE SPRITE"
61040 PRINTTAB(4)"<DCRSR>SUBROUTINE FOR
       YOUR PROGRAM?"
61045 PRINTTAB(4)"<RED><DCRSR><RVSON>WA
      RNING<RVSOF>:<BLU> IF YOUR REPLY
      IS 'Y'"
61050 PRINTTAB(4)"THEN SPRITE.GEN WILL
      BE DELETED"
61060 INPUT"<DCRSR><4RCRSR>CREATE SUBRO
      UTINE (Y/N)   <3LCRSR>";A$:IFA$="
      N"THENEND
61070 IFA$<>"Y"THENPRINT"<2UCRSR>":GOTO
      61060
61080 PRINTTAB(4)"<2DCRSR><BRWN>HOW MAN
      Y SPRITES HAVE YOU":INPUT"<DCRSR>
      <6RCRSR>DESIGNED    <4LCRSR>";N
62000 PRINT"<CLR><2DCRSR>"
62010 PRINT"10 PRINT";CHR$(34);"<CLR><B
      LU>";CHR$(34)
62020 PRINT"15 N=";N
62030 PRINT"20 GOSUB 50000":PRINT"GOTO
      62050"
62040 PRINT"<HOME>": FOR X=631 TO 637:P
      OKE X,13:NEXT:POKE 198,6:END
62050 X=2049
62060 IF PEEK(X)=58 AND PEEK(X+1)=64 TH
      EN 62080
62070 X=X+1:PRINT"<HOME><13DCRSR><4RCRS
      R>";X:GOTO 62060
62080 POKE X-5,0
62090 POKE X-4,0
62100 END
```

פרק 5
הספרייטים משתלבים במשחקי מחשב

השימוש ב–Sprite.Gen

בפרק זה נפתח משחק הקרוי 'מטרה' אשר משתמש ב-SPRITE.GEN וחוקר את אפשרויותיה השונות.

מטרה: המשחק

המשימה ב"מטרה" היא לפגוע במטרה במרכז המסך עם טיל סיבובי אשר נשלט בשלושה צירים על-ידי לחיצת מקשים מסוימים. שני מקשים שולטים בכיווני X ו-Y (Z ו-X בהתאמה) ואילו שני מקשי הסמן ללא SHIFT שולטים בכיוון Z. מקש הסמן <—> מניע את הטיל לתוך המסך ואילו מקש הסמן מניע אותו לכיוון השחקן.
מכיון שהמטרה נתונה בחלל, הטיל יכול לעבור מתחת או מעל המטרה וכדי להשיג פגיעה על הטיל להיות בגובה הנכון. בזמן נסיונו לפגוע במטרה על השחקן להמנע ממכשולים על המסך. התנגשות עם אחד מאלו תסיים את המשחק כפי שתעשה גם פגיעה במטרה. בסיום תדווח התקדמות השחקן יחד עם זמן הריצה של המשחק.

כיצד זה עובד...

ציר Z דמיוני עבור המטרה נוצר באמצעות פונקצית מספר אקראי. כאשר מתעדכן ציר Z של הטיל הוא עלול להמצא במקום הזהה לזה של המטרה או לפניה או מאחוריה. כאשר צירי X, Y ו-Z של הטיל והמטרה זהים נרשמת פגיעה וסבוב זה של המשחק מסתיים עם הודעה מתאימה. כאשר ציר Z של הטיל הוא לפני המטרה הטיל חייב לעבור לפניה, וכאשר הציר נמצא מאחורי המטרה, המטרה חייבת להסתיר את הטיל בזמן מעברו.

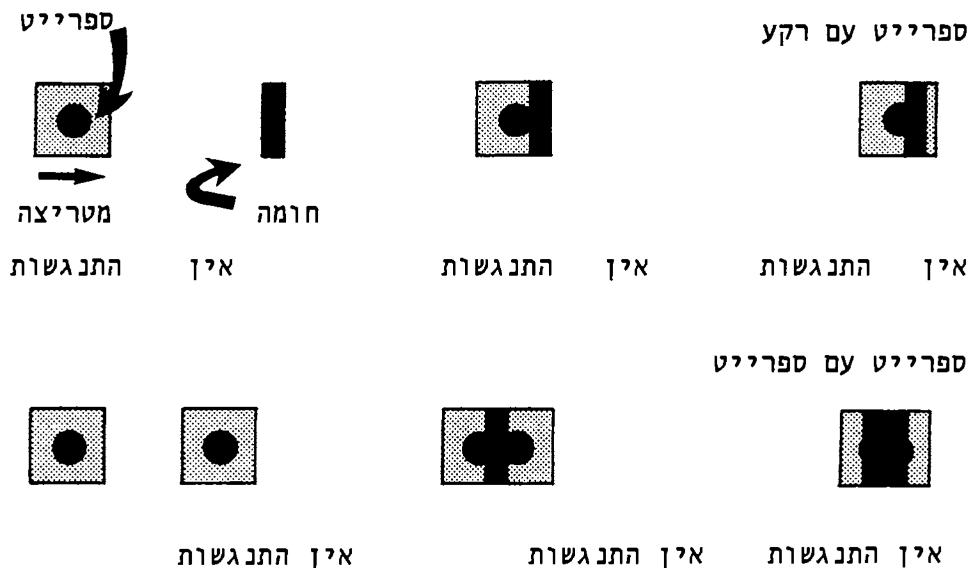
קדימויות הספרייטים

אפקט התלת-מימדיות מושג בקלות על-ידי שמוש בתכונת הקדימויות של הספרייטים אשר מגדירה את הסדר בו מופיעים הספרייטים כאשר הם באים זה לפני זה. בסדר המקורי של הקומודור יופיע ספרייט 0 תמיד לפני כל האחרים וספרייט 7 תמיד מתחבא מאחור. מה שדרוש למעשה במשחק זה הוא ספרייט אחד - הטיל - אשר עובר לפעמים לפני ולפעמים מאחורי ספרייט שני. לא קל להשיג זאת ולכן משתמשים בשלושה ספרייטים, 0 ו-2 עבור המטרה ו-1 עבור הטיל. שתי המטרות יכולות להדלק או להכבות כדי לאפשר לטיל לעבור לפניהן או אחריהן לפי הצורך.

התנגשויות של הספרייטים

אך ראשית: מהי התנגשות?

ספרייט מורכב ממטריצת מלבן של 24*21 נקודות והתנגשות מתרחשת רק כאשר מלבן זה בא במגע עם מלבן דומה של ספרייט אחר או עם קיומו של עצם במסך. ציור 5.1 ממחיש זאת:



ציור 5.1

שים לב - התנגשות מתרחשת רק כאשר גוף או עיקר הספרייט נמצא במגע עם זה של ספרייט אחר.

(i) התנגשות עם ספרייטים אחרים

בית אחד ברגיסטר הספרייטים, 53278 (R+30) משמש למעקב אחרי התנגשויות ספרייט/ספרייט. וכך כאשר ספרייט אפס מתנגש עם ספרייט אחר, סיבית אפס של R+30 נקבעת ל-1. וגם כמובן נקבעת הסיבית המתאימה של הספרייט האחר. לפיכך קריאת R+30 תגיד אם התרחשה התנגשות ספרייט עם ספרייט, אך זהירות! קריאת כתובת זו מאפסת את הבית חזרה לאפס כך שעליך לתפוס אותו בפעם הראשונה כאשר משתמשים רק בשני ספרייטים. בדיקת התנגשות זו היא קלה מכיון שקריאת 53278 תראה משהו שונה מאפס כאשר ארעה התנגשות. אם נערכת בדיקה עבור התנגשות מסוימת הרי שתוכן הרגיסטר חייב להבחן יותר לעומק. למשל - לאחר התנגשות בין ספרייט 5 לספרייט 1, סיביות 1 ו-5 יקבעו ל-1. וכך יכיל הרגיסטר 2#1 ועוד 2#5 - כלומר:

7	6	5	4	3	2	1	0	
0	0	1	0	0	0	1	0	
0 +	0 +	32 +	0 +	0 +	0 +	2 +	0	= 34

קריאת 53278 (R+30) תהיה 34. דרך מתוחכמת יותר לבדיקה זו תהיה OR בין תוכן הקריאה לבין הכמות המתאימה - במקרה זה 34. כאשר מבצעים זו - האחדים בסיבית הראשונה והחמישית ישארו יחד עם אלו של הבית המקורי. וכך אם תוצאת ה-OR בין הבית ו-34 היא 34 הרי שזה מה שהכיל הבית המקורי. כלומר:

OR עם	0	1	0	0	0	1	0	0
	0	1	0	0	0	1	0	0
נותן	0	1	0	0	0	1	0	0
=	0 +	2 +	0 +	0 +	0 +	32 +	0 +	0 = 34

אך

```
        0   1   0   0   1   1   1   0}   OR עם
        0   0   1   0   0   0   1   0}
        0   1   1   0   1   1   1   0    נותן
110  =  0 +64 +32 + 0 + 8 + 4 + 2 + 0    =
```

אם צריך לבדוק יותר מהתנגשות אחת, הרי שתוכן קריאת הרגיסטר חייב להיות מאוחסן לפני או בזמן הבדיקה הראשונה, אחרת יהרס תוכן הרגיסטר.

(ii) התנגשות עם עצמים אחרים

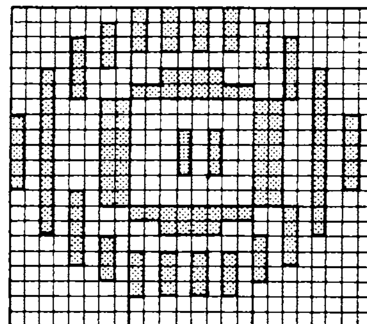
אם ארעה התנגשות בין ספרייט למשהו בצבע השונה מצבע הרקע, רגיסטר R+31 (53279) נקבע בהתאם. הפעולה היא למעשה זהה להתנגשויות ספרייט עם ספרייט והיא נבחנת באותה דרך. וכך לבדוק התנגשות של ספרייט 4, R+31 (53279) יקרא כדי לראות אם סיבית 4 נקבעה. כלומר:

IF PEEK (53279) AND 24 = 24 THEN - התנגשות

שים לב כי שורה זו תבחון רק את התנגשות ספרייט 4 ותהרוס את ההוכחה של כל התנגשות אחרת שארעה.

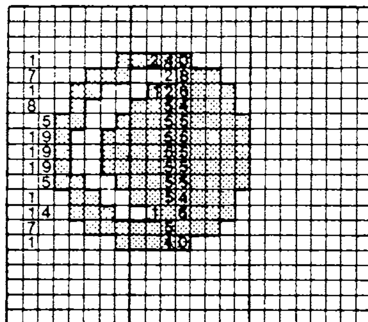
פיתוח התוכנית

מכיון שהתוכנית עוסקת כולה בספרייטים, המשימה הראשונה לביצוע היא עיצוב הטיל והמטרה בעזרת SPRITE.GEN. ציור 5.2 נותן את העיצוב המשמש למשחק מטרה. אך מכיון שכנראה תמצא את העצוב לא לטעמך, אתה חופשי לעצב לפי טעמך. זהו אחד מהיתרונות הגדולים של ספרייטים, לאחר שהאלגוריתם לטיפול בהם מפותח, פעולתו אינה תלויה בצורתם.



C0=15 C1=10 C2=7

מטרה



טיל

ציור 5.2

כדי ליצור צורות אלו, ראשית טען והרץ את SPRITE.GEN (אם היא עדיין לא בזכרון).

ראשית הטיל...

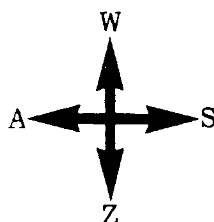
הקומודור 64 שואל:

ENLARGE X? N השב

ENLARGE Y? N השב

MULTICOLOR? N השב

אחרי כן יציג המחשב את רשת עצוב הספרייט ויחכה לך. בשלב זה ממוקם סמן העיצוב בפינה השמאלית-עליונה של הרשת והוא נע בעזרת המקשים:



מקשי הפונקציה פועלים באותה צורה כמו אלו ב-CHAR.GEN עם תוספת F6.

F1 - הדלק את הפיקסל העכשווי.

F3 - כבה את הפיקסל העכשווי.

F5 - הדפס את ה-DATA על המסך.

F6 - הוסף את ה-DATA לתוכנית.

F7 - נקה את מטריצת העצוב.

הכנסת מידע העיצוב עצמו נעשית בדיוק כמו ב-CHAR.GEN כך שאין צורך בהסברים נוספים.

כאשר העיצוב מוכן, לחץ על מקש פונקציה 6, המושג על-ידי לחיצה על SHIFT ולחיצה על מקש הפונקציה המסומן F6/F5.

אחרי-כן תשאל:

AT WHICH LINE DO YOU WANT TO STORE YOUR SPRITE DATA (1000-5000)?

השב - 1000

השאלה הבאה:

IN STEPS OF?   השב - 10

WHICH SPRITE WILL THIS BE (0-7)?   השב - 1

AT WHICH BLOCK DO YOU WANT THE DATA TO BE STORED?   השב - 14

אחרי-כן תציג התוכנית את השורות הנוספות על המסך, תכניס אותם אל התוכנית ותציג:

PRESS ANY KEY TO CONTINUE

עם לחיצת מקש, תוכנית SPRITE.GEN תשאל אותך:

ARE YOU FINISHED USING SPRITE.GEN (Y/N)?

השב ב-N ותוכנית SPRITE.GEN תתחיל מחדש ותאפשר כעת את עיצובה של המטרה.

חזור על התהליך למעלה עבור המטרה אך השב כך:

השב N	ENLARGED X ?
השב N	ENLARGED Y ?
השב Y	MULTICOLOUR ?
השב 10	MULTI-COLOUR# 1
השב 15	SPRITE COLOUR
השב 7	MULTICOLOUR#2

כעת תכנן את הספרייט; הקש F6 וענה לשאלות כדלקמן:

```
STORE AT LINE?        השב 10070
IN STEPS OF?          השב 10
WHICH SPRITE?         השב 0
BLOCK STORED?         השב 13
```

ואז...

מכיון ששתי המטרות זהות, העיצוב של הראשונה יכול לשמש עבור השניה וכך יש צורך רק לספר למחשב להתבונן באותו הזכרון עבור כל ספרייט. דבר זה נעשה בתחילת התוכנית עם POKE ישיר. ברור שאם ברצונך לעבור על התהליך במדויק אתה יכול לחזור על הפרוצדורה למעלה.

POKE 2042, 13

לקטע 13 — כלומר קבע את ספרייט 2

בשלב זה, אם אתה מרוצה מהספרייטים שלך, אתה יכול לבחור ב-RETURN בזמן היותך במצב עריכה ואם הצלחת לשכנע את המחשב שאתה רציני, מחק את SPRITE.GEN והשאר את ה-DATA ואת שגרת יצירת הספרייטים.

וסוף סוף - המשחק ...

עקרונית - התוכנית מתחלקת לחמישה חלקים:

1 - הכנה

2 - אלגוריתם המשחק

3 - דיווח על התקדמות

4 - הדפסת מכשולים על המסך

5 - DATA ובנית הספרייטים

חלק 1: הכנה

ודאי תרצה כרגע לטעון את תוכנית "מטרה" ולעקוב אחר תכנון המשחק. תוכל לבצע LIST של חלקי התוכנית כאשר נעסוק בהן (כלומר - LIST 100-160).

חלק זה מתחיל בשגרות הכנת הספרייטים הרגילות, תוכנית 5.1 מראה שגרות אלו ביחד עם הסברים על פעולתן.

100 POKE 53269,0:	כבוי כל הספרייטים
PRINT"<CLR><NOCBM><UCASE>";:	נקוי המסך
GOSUB 50000:	קריאת ה-DATA ויצירת הספרייטים חלק 5
POKE 2042,13	קבע את ספרייט 2 כזהה לספרייט 0
110 POKE 53280,3:POKE 53281,1	קבע צבע מסך
130 POKE 53289,15:	קבע את צבעיו הנפרדים של ספרייט 2
POKE 53276,5	קבע מצב רב-צבעוני עבור ספרייט 0 וספרייט 2
140 POKE 53248,170:POKE 53252,170	קבע קואורדינטות X עבור ספרייטים 0 ו-2
150 POKE 53249,100:POKE 53253,100	קבע קואורדינטות Y עבור ספרייטים 0 ו-2
160 GOSUB 820	הדפס מכשולים

אחרי ביצוע הכנת הספרייטים, הקואורדינטות ההתחלתיות של הטיל יכולות להקבע. שורה 240 עושה זאת עם מתן ערך פשוט, x=24:y=60. יש צורך במשחק זה ביכולת לשנות את כיוון הטיל כאשר הוא נשלט על-ידי מקש, או כשהוא פוגע במגן האלקטרו-מגנטי סביב שדה המשחק. דבר זה נעשה באמצעות "צמד העלאות", XD ו-YD אשר נקבעים לקבל ערכים שליליים או חיוביים. בזמן ההכנה נקבעים שני אלו ל-5 אשר נותן את מספר הפיקסלים להעלאה עבור כל סבוב. שנוי ערך זה ישנה את מהירות הטיל על המסך - ערך של 1 יתן התקדמות איטית יותר ו-10 - מהירה יותר.

בנוסף לקואורדינטות X ו-Y, גם הטיל וגם המטרה זקוקים לערך Z או עומק. דבר זה נעשה באמצעות שתי פונקציות מספר אקראי אשר מחשבות את L, קואורדינטת Z של הטיל, ואת RL, קואורדינטת Z של המטרה.

בדיקה פשוטה בשורה 250, תוכנית 5.2 שולחת את התוכנית אחורה לנסות שנית כאשר שתי הקואורדינטות ה-Z זהות.

כלומר:

תוכנית 5.2

```
240 X=24:Y=60:XD=5:YD=5:X1=0:X2=0:L=INT(
RND(0)*10):RL=INT(RND(1)*5)+3
250 IF L=RL THEN 240
```

לבסוף מאופסים שני רגיסטרי ההתנגשות על-ידי קריאתם. השעון (TI$) נקבע לאפס והחזרה האוטומטית מבוטלת. כלומר:

תוכנית 5.3

```
260 CS=PEEK(53278):CD=PEEK(53279):
TI$="000000":POKE 650,64
```

חלק 2: אלגוריתם המשחק

בתחילת המשחק מוגדלות קואורדינטות X ו-Y ב-5 (XD ו-YD), דבר היוצר תנועה לרוחב המסך ב-45 מעלות. לפני שאלו מאוחסנות למעשה פנימה כדי למקם מחדש את הטיל, הוא מכובה על-ידי POKE. כאשר דבר זה נעשה, מבוצע POKE של קואורדינטות הספרייט החדשות לזכרון לפני הפעלתו מחדש של הספרייט. קואורדינטת X מוזרה במקצת מכיון שיכולתה להיות גדולה מ-255 צריכה להחזות מראש.
לפיכך, הקואורדינטה עצמה צריכה להתחלק לחלקיה היותר ופחות משמעותיים. ראשית - כדי למצוא את הבית הפחות משמעותי, בצע AND עם 255. נסה זאת לדוגמא עם 500:

בפעולת AND	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	= 500
עם	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	= 255
נותן	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	= 244

כלומר: ה-POKE נעשה

```
POKE 53250,X AND 255
```

כדי להשיג את הבית היותר משמעותי, מתבצע AND של קואורדינטת X עם 256 כדי לבדוק בקואורדינטה גדולה מ-255. מכיון שב-256 סיבית שמונה קבועה ל-1 ושאר הסיביות קבועות לאפס, הדבר מסלק כל סיבית קבועה מבין סיביות 0 עד 7 ומשאירה כל 1 אם הוא קבוע בסיבית שמונה. נסה זאת עם 500.

	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0	1	= 500
בפעולה AND עם	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	= 256
נותן	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	= 256

כאשר 1 - כלומר 256 - נמצא, הוא חייב להפוך ל-1 בסיבית המתאימה לספרייט עצמו. מכיון שכתובת 53264 מכילה אינפורמצית MSB עבור כל שמונת הספרייטים. עבור ספרייט 0, ה-POKE המתאים יהיה 1 או 2↑0, ואילו עבור ספרייט 4 יהיה הערך 16 או 2↑4. וכך ה-256 צריך להיות מחולק ב-256 כדי להשיג את ערך ה-POKE עבור ספרייט אפס או ב-2 כדי להשיג את הערך עבור ספרייט 7. בצורה כללית המחולק הוא (מספר הספרייט - 8)↑2. במקרה הפרטי של ספרייט 1, ה-256 צריך להתחלק ב-128 כדי לתת 2. אם נבצע זאת נקבל שורה:

```
POKE 53264,(X AND 256)/128
```

תוכנית 5.4

270 X=X+XD:Y=Y+YD	הגדל את X ו-Y
280 POKE 53250, X AND 255: POKE 53251, Y:POKE 53264, (X AND 256)/128	קבע X חדש וקואורדינטת Y
POKE 53269,N+2	הדלק שוב את ספרייט הטיל

כאשר הטיל חוצה את המסך, חייבת להערך בדיקה כדי לראות אם הוא הגיע לגבולות מסעו. אמצעי רגיל למדי מסדר זאת, כאשר מגיע הטיל לגבולות המסך, הוא מופנה חזרה אל מרכז המסך (קראנו לזה "ניתור" במשחק אחר!) כפי שזה היה במשחק הכדור, דבר זה נעשה על-ידי הפוך סימן ההגדלה בשורות 290 ו-300.

תוכנית 5.5

```
290 IF X>324 OR X<24 THEN XD=-XD
300 IF Y>230 OR Y<60 THEN YD=-YD
310 IF L<5 THEN POKE 53269,6:N=4
320 IF L>=5 THEN POKE 53269,3:N=1
```

לאחר מכן בא GET A$ אשר בודק את מחסנית המקלדת. המחסנית מאופסת עם POKE 198,0 ושני רגיסטרי ההתנגשות של הספרייטים מאוחסנים כ-CS (התנגשות בספרייטים) ו-CD (התנגשות ברקע). אחת מן הבעיות במשחק המשתמש בשלושה צירים, היא כיצד להציג את הציר השלישי, תנועה בציר Z שהיא תנועה לפנים ומחוץ למסך אינה נראית למעשה על-ידי השחקן. כדי להתגבר על בעיה זו, נוספות למשחק שתי תכונות נוספות כדי לציין זאת.

א - תנועה התקיימה בציר Z

ב - הושג הגבול של תנועה בציר זה.

צורת המראה שנבחרה היא הבהוב צבע הרקע עבור תנועה והבהוב משולש עבור השגת גבול התנועה. לשתים מבין שורות אלו יש את המבנה הבסיסי:

"אם GET A$ היה סמן-מטה (<DCRSR>) וקואורדינטת Z גדולה מאפס (Z>0) ולא נעשתה כל התנגשות ספרייט בספרייט (CS=0) הרי מופחתת קואורדינטת Z (L=L-1) והבהוב הגבול (POKE 53281,0:POKE 53281,1)" - כלומר:

תוכנית 5.6

```
330 GETA$
340 POKE198,0:CS=PEEK(53278):CD=PEEK(53
    279)
350 IFA$="<DCRSR>"ANDL>0ANDCS=0THENL=L-
    1:POKE53281,0:POKE53281,1
```

ובדומה עבור העבר השני:

תוכנית 5.6(a)

```
360 IFA$="<RCRSR>"ANDL<10ANDCS=0THENL=L
    +1:POKE53281,0:POKE53281,1
370 IF(A$="<DCRSR>"ANDL=0)OR(A$="<RCRSR
    >"ANDL=10)THENGOSUB810
```

אחרי-כן באה בדיקה עבור התנגשות, אשר בנקודה זו קואורדינטות 2 של הטיל והמטרה יתאימו בדיוק (Z=RL) ורגיסטר ההתנגשות של הספרייטים יקבע (CS>0). כאשר נמצא תנאי זה, התוכנית מכוונת לשגרה ב-580.

תוכנית 5.6(b)

```
380 IF L=RL AND CS>0 THEN A$=TI$:GOTO 580
```

בזמן שהספרייט נע על המסך, כיוונו יכול להשתנות באמצעות מקשי 'Z' ו-'X' אשר למעשה יוצרים קיר בלתי נראה אשר ממנו הטיל מנתר. התחביר של השורה הוא די ישיר: אמצעי הזהירות היחיד הנדרש הוא בדיקה שהטיל עדיין לא הגיע לאחד מגבולות השדה. כאשר דבר זה קורה בסבוב הנוכחי, כיוון התנועה כבר שונה ושנוי נוסף יהיה הרסני!

תוכנית 5.7

```
390 IF A$="Z"AND X<324 AND X>24 THEN XD=-XD
400 IF A$="X"AND Y<230 AND Y>60 THEN YD=-YD
```

ולבסוף, אלגוריתם המשחק מגיע עד כדי בדיקה של התנגשות בין הספרייט לרקע. למרות שבדיקה לערך שונה מאפס תספיק, נשתמש ב-AND כדי להדגים כיצד לבדוק עבור מידע ספרייטים ספציפי. כאשר מבחינים בהתנגשות בין ספרייט לרקע התוכנית נשלחת לשגרה ב-450 אשר מודיעה בהתאם.

תוכנית 5.8

```
420 IF (CD AND 2)=2 THEN 450
```

דבר זה אינו שווה מאומה כאשר ספרייטים אחדים מעורבים בענין, אחסון ה-PEEK כמשתנה הוא נחוץ ביותר. PEEK של רגיסטר התנגשות בספרייטים הורס את המידע. דבר זה <u>אינו</u> כמובן תופס עבור כתובות אחרות למעט הרגיסטרים האחרונים של SID.
ולבסוף, בסיומו של סבוב ללא פגיעה, הזמן TI$ מדווח והתוכנית חוזרת לסבוב נוסף.

תוכנית 5.9

```
430 PRINT"<HOME><RVSON>";TI$
440 GOTO 270
```

חלק 3: דיווח על התקדמות

דיווח על טיל מתפוצץ/מחטיא

כאשר מתרחשת התנגשות, המשימה הראשונה היא אחסון הזמן, TI$, כדי לראות כמה זמן ארך המשחק - כלומר A$=TI$.

ברגע שהטיל פגע בעצם יש לפוצצו. אפקט כזה יכול להווצר על-ידי הגדרת מספר טילים מפוררים, או בקלות יותר, על-ידי ציור פשוט של רסיסי עצמים אקראיים בקטע המסך של הספרייט.

מכיון ששבב ה-Vic יודע היכן נמצא מידע הספרייט באמצעות מספרי הקטעים השמורים בכתובת 2040 עד 2047, הכתובת הספציפית עבור הספרייט המפורר יכולה להשתנות במהירות, דבר השולח את שבב ה-Vic לחפש את המידע בכל מקום.כלומר:

```
FOR X = 16 TO 0 STEP-1:POKE 2041,X:NEXT
```

דבר זה אומר לשבב ה-Vic כי המידע עבור ספרייט 1 נמצא בקטע 16, ואחר-כך בטקע 15 וכו' עד אפס. אם השגרה כלולה בלולאה אחרת, נגיד בלולאה בת ארבעה מחזורים, הרי שהאפקט ימשך זמן סביר. סכנה אחת הקשורה בטכניקה זו היא שהיא משאירה את ספרייט 1 בקטע אפס - שהוא דף אפס (ZERO PAGE). אך כאשר התוכנית מתחילה מחדש עם 'RUN' (שורה 730) ממוקם מחדש על-ידי התוכנית.

לפני הצגת הודעת סוף המשחק, הטיל מכובה (POKE 53269,N). ולבסוף מכובית המטרה ומתבצע נקוי מסך. כלומר - תוכנית 5.10

תוכנית 5.10

450 A$=TI$:	זכור את הזמן
FOR Y=1 TO 4:FOR X=16 TO 0 STEP -1:	צייר ספרייטים אקראיים
POKE 2041,X:NEXT:NEXT	4x16 פעמים
460 FOR X=1 TO 1000:NEXT:	עכוב
POKE 53269,N	כבה את הטיל
470 FOR X=1 TO 1000:NEXT:	עכוב
POKE 53269,0:PRINT"<CLR>";	כבה את המטרה

אחרי-כן מדווחת התוכנית כמה זמן הצליח השחקן לשרוד באמצעות השגרה ב-510. כמו-כן היא מספרת לשחקן(ית) כי הוא נכשל בפגיעה במטרה. לאחר מכן התוכנית קופצת לשגרת
'DO YOU WANT ANOTHER GO?'
כלומר:

תוכנית 5.11

```
480 PRINT "<CLR><7DCRSR><9RCRSR> <2RCRS
R><ORNG>YOU SURVIVED FOR :":GOSUB 510
490 PRINT "<2DCRSR><5RCRSR><GRN>AND YOU
DID NOT HIT THE TARGET!"
500 GOTO 690
```

דיווח על הזמן

הזמן במשחק זה מאוחסן ב-TI$ אשר אופס בתחילתו של מחזור המשחק (שורה 260) ואשר אוחסן ב-A$ כאשר המשחק הופסק. לפני הצגת הזמן, המחרוזת TI$ חייבת להחתך כדי לקבל את השעות, הדקות והשניות, כלומר:

שתי הספרות השמאליות ביותר - שעות LEFT$(A$,2)
שתי הספרות האמצעיות - דקות MID$(A$,3,2)
שתי הספרות הימניות ביותר - שניות RIGHT$(A$,2)

פענוחה של המחרוזת יכול להעשות בצורה ישירה במשפט המודפס כמו בתוכנית 5.12, שורות 510 עד 530.

תוכנית 5.12

```
510 PRINT "<DCRSR><17RCRSR><BLU>;LEFT$(A
$,2);" HOURS<DCRSR><8LRCRSR>";
520 PRINT MID$(A$,3,2)" MINUTES
<DCRSR><14LRCRSR> AND ";
530 PRINT RIGHT$(A$,2);" SECONDS"
```

אחר ביצוע הדיווח מודפסת הודעת המשחק על המסך והשגרה חוזרת.

תוכנית 5.13

```
540 PRINT "<HOME><2DCRSR>       ";
TAB(16);"<RSVON><RED>        "
550 PRINT TAB(16);"<RVSON> TARGET
<RVSOFF>"
560 PRINT TAB(16);"<RVSON>
<RVSOFF>"
570 PRINT "<10DCRSR>";:RETURN
```

דיווח על פגיעה

ההמשך של חלק זה עוסק בהדפסת הודעות על המסך אחרי ש-TI$ אוחסן ב-A$. קריאה לשגרה ב-510 מציגה את הזמן. הודעה מיוחדת כעת נבחרת מתוך המבחר השמור בשורות 630 עד 660, הבחירה הפרטית מבוססת על הזמן האמיתי שלקח לפגוע במטרה. כדי לבצע זאת, התוכנית עוברת דרך בדיקות IF...THEN ואם המבחן מצליח, היא מדפיסה את ההודעה המתאימה ואחר היא מכוונת אל שגרת "DO YOU WANT ANOTHER GO", כלומר:

תוכנית 5.14

```
580 FORX=1TO50:POKE53269,3:POKE53269,1:
    NEXT
590 FORX=1TO1000:NEXT:POKE53269,0
600 PRINT"<CLR><7DCRSR><LTBLU>"TAB(16);
    "WELL DONE!":PRINTTAB(10);" YOU HIT
     THE TARGET."
610 PRINT"<DCRSR><14RCRSR><BLK>YOUR TIM
    E WAS:":GOSUB510
620 PRINT"<DCRSR><5RCRSR><BRWN>WHICH WA
    S ";
630 IFA$<"000010"THENPRINT"BRILLIANT! F
    ANTASTIC! AMAZING!":GOTO690
640 IFA$<"000030"THENPRINT"BRILLIANT":G
    OTO690
650 IFA$<"000100"THENPRINT"VERY GOOD":G
    OTO690
660 IFA$<"001000"THENPRINT"NOT AT ALL G
    OOD":GOTO690
670 PRINT"NOT VERY GOOD BUT AT LEAST YO
    U HIT IT. JUST KEEP ON"
680 PRINT"YOU'LL GET A BETTER SCORE SOO
    N."
```

רוצה משחק נוסף ? Do you want another go

זוהי שגרה רגילה למדי. היא מחכה לתשובת Y או N ובשעת קבלת Y היא מריצה מחדש את התוכנית באמצעות פקודת RUN בשורה 730. אחד מיתרונותיו של RUN על 100 GOTO (בתוכנית זו) הוא שכשמערכת ההפעלה רואה את הפקודה RUN, היא אוטומטית מאפסת את כל המשתנים, דבר המבטיח כי המשחק יתחיל פעם נוספת עם גליון נקי. תוכנית 5.15 מראה את מלוא השגרה.

תוכנית 5.15

```
690 PRINT "<4DCRSR><8RCRSR><ORNG>
DO YOU WNAT ANOTHER GO?"
700 PRINT "<DCRSR><15RCRSR><ORNG>Y
<LTRED>ES OR <ORNG><LTRED>0"
710 GETC$:IFC$=""THEN 760
720 IF C$="N" THEN 750
730 IF C$="Y" THEN RUN
740 GOTO 710
```

סיום

כאשר השחקן מציין כי הוא סיים את המשחק, מוצג מסך המטרה וניתנת הודעה ידידותית. זו מוחזקת על המסך לזמן-מה עם אמצעים להפסקת התהליך על-ידי השחקן אם רצונו בכך.

תוכנית 5.16

```
760 PRINT"<CLR>":GOSUB540:PRINTTAB(12)"
    <2DCRSR><PUR>GOODBYE FOR NOW."
770 FORX=1TO1000:GETA$:IFA$=""THENNEXT
780 IFX<1000THENX=1010:NEXT
790 PRINT"<CLR>";
800 POKE650,0:PRINT"<OKCBM>":END
```

מחזור הצבע

זוהי שגרה אשר יוצרת מחזור דרך הצבעים 0 ל-10 כאשר נעשה נסיון לזוז בציר Z מעבר לגבול.

תוכנית 5.17

```
810 FOR Z=0 TO 10:POKE 53281,Z:
NEXT:POKE 53281,1:RETURN
```

חלק 4: הצגת המכשולים

לא יעשה נסיון להסביר חלק זה, משום שהוא פשוט מדפיס תוים על המסך. זהו בברור יתרון חשוב של ספרייטים שכה קל לאתר התנגשויות ודבר זה מאפשר להדפיס מבחן מכשולים שיתאימו לטעמו של כל אחד.

תוכנית 5.18

```
820 PRINT"<CLR><16DCRSR><29RCRSR><YEL>V
    <LTRED>*4<YEL>V";
830 PRINT"<DCRSR><4LCRSR><LTRED>/<CYN>U
    I<LTRED>/";:PRINT"<DCRSR><4LCRSR><L
    TRED>7<CYN>JK<LTRED>7";
840 PRINT"<DCRSR><4LCRSR><YEL>V<LTRED>*
    4<YEL>V";
850 PRINT"<4DCRSR><LCRSR><GRY2>V<UCRSR>
    V<UCRSR>V<UCRSR>V<UCRSR>V<UCRSR>V<U
    CRSR>V<UCRSR>";
860 PRINT"<BLU><HOME><RVSON>
                                 <RVSOF
    >";
870 PRINT"<HOME><16DCRSR><7RCRSR><YEL>V
    <LTRED>*4<YEL>V";
880 PRINT"<DCRSR><4LCRSR><LTRED>/<CYN>U
    I<LTRED>/";:PRINT"<DCRSR><4LCRSR><L
    TRED>7<CYN>JK<LTRED>7";
890 PRINT"<DCRSR><4LCRSR><YEL>V<LTRED>*
    4<YEL>V";
900 PRINT"<2UCRSR><10LCRSR><GRY2>V<DCRS
    R>V<DCRSR>V<DCRSR>V<DCRSR>V<DCRSR>V
    <DCRSR>V<DCRSR><BLU>";
910 RETURN
```

אם תמצא את המבחר כקבוצה משעממת של מכשולים, פשוט שנה את משפטי ה-PRINT כדי שיתאימו לך. מכונת איתור ההתנגשויות של הספרייטים תנהל את בעית ההתנגשות אוטומטית.

תכונה זו נותנת לך שליטה חופשית בעיצוב המסך שלך. תוכל לעצבו כמבוך או בכל צורה. זה תלוי בך!

מעשיה האחרונים של SPRITE.GEN לפני הריסתה חסרת הרחמים היו להדפיס את משפטי ה-DATA של הספרייטים. כל מה שנותר לעשות לאחר לחיצת הכפתור הוא לקבוע את הספרייטים ודבר זה הוסבר ב-SPRITE.GEN. לפיכך - שוב - אין הסברים.

תוכנית 5.19

```
10000 DATA1,14,6,0,0,0,0,0,0,0
10010 DATA0,0,0,0,0,0,0,1,240,0
10020 DATA7,28,0,12,126,0,8,254,0,25
10030 DATA255,0,19,255,0,19,255,0,19,25
      5
10040 DATA0,25,255,0,12,254,0,14,126,0
10050 DATA7,252,0,1,240,0,0,0,0,0
10060 DATA0,0,0,0,0,0,0,0,0, 0
10070 DATA0,13,15,0,0,1,10,7,0,170
10080 DATA0,2,170,128,10,170,160,10,0,1
      60
10090 DATA40,60,40,40,255,40,35,0,200,1
      63
10100 DATA0,202,163,20,202,163,20,202,1
      63,20
10110 DATA202,163,0,202,35,0,200,40,255
      ,40
10120 DATA40,60,40,10,0,160,10,170,160,
      2
10130 DATA170,128,0,170,0,0,0,0,0,0, 0
50000 CN= 2
50010 POKE53277,0:POKE53271,0:POKE53276

50020 FORY=1TOCN:READSN,BN,CO,EX,EY,MC,

50030 READC:POKE(BN*64)+X,C
50040 NEXT
50050 POKE2040+SN,BN
50060 POKE53287+SN,CO
50070 POKE53277,PEEK(53277)OR((2↑SN)*EX

50080 POKE53271,PEEK(53271)OR((2↑SN)*EY

50090 POKE53276,PEEK(53276)OR((2↑SN)*MC

50100 IFMC=1THENPOKE53285,C1:POKE53286,

50110 NEXT:RETURN
```

כעת המשחק גמור. רק נחזור על הכללים:

Z ו-X מזיזים את הטיל ומקפיצים אותו חזרה בכיווני X ו-Y.

מקשי הסמן מזיזים את הטיל לתוך ומחוץ למסך, כלומר בכיוון Z.

⇕ מניע את הטיל מחוץ למסך.

⇔ מניע את הטיל לתוך המסך.

זהו. כולו שלך. משחק מתסכל מעט, לא?

נספח א
קבוצת הסימנים המיוחדת של ד"ר ווטסון

סימול	מתקבל ע"י	משמעות
<UCRSR>	(<SHIFT><CRSR ⇑ >)	מסמן למעלה
<DCRSR>	(<CRSR ⇓ >)	מסמן למטה
<RCRSR>	(<CRSR ⇒>)	מסמן ימינה
<LCRSR>	(<SHIFT><CRSR ⇐ >)	מסמן שמאלה
<CLR>	(<SHIFT><CLR/HOME>)	נקוי מסך
<HOME>	(<CLR/HOME>)	HOME
<RVS OFF>	(<CTRL>0)	כבוי נגטיבי
<RVS ON>	(<CTRL>9)	הדלקת נגטיבי
<UCASE>	(<SHIFT><CBM>)	אותיות גדולות
<LCASE>	(<CTRL>N)or<SHIFT> <CBM>)	אותיות קטנות
<NOCBM>	(<CTRL>H)	לא לאפשר CBM
(OKCBM>	(<CTRL>I)	לא לאפשר CBM
<BLK>	(<CTRL><1>)	שחור
<WHT>	(<CTRL><2>)	לבן
<RED>	(<CTRL><3>)	אדום
<CYN>	(<CTRL><4>)	טורקיז
<PUR>	(<CTRL><5>)	סגול
<GRN>	(<CTRL><6>)	ירוק
<BLU>	(<CTRL><7>)	כחול
<YEL>	(<CTRL><8>)	צהוב
<ORNG>	(<CBM><1>)	כתום
<BRWN>	(<CBM><2>)	חום
<LTRED>	(<CBM><3>)	אדום בהיר
<GRY1>	(<CBM><4>)	אפור 1
<GRY2>	(<CBM><5>)	אפור 2
<LTGRN>	(<CBM><6>)	ירוק בהיר
<LTBLU>	(<CBM><7>)	כחול בהיר
<GRY3>	(<CBM><8>)	אפור 3

נספח ב
קבוצת התווים של הקומודור 64

טבלה 1

poke	set1	set2	poke	set1	set2	poke	set1	set2	poke	set1	set2
0	@		32	<space>		64			96	<space>	
1	A	a	33	!		65		A	97		
2	B	b	34	"		66		B	98		
3	C	c	35	#		67		C	99		
4	D	d	36	$		68		D	100		
5	E	e	37	%		69		E	101		
6	F	f	38	&		70		F	102		
7	G	g	39	'		71		G	103		
8	H	h	40	(		72		H	104		
9	I	i	41	)		73		I	105		
10	J	j	42	*		74		J	106		
11	K	k	43	+		75		K	107		
12	L	l	44	,		76		L	108		
13	M	m	45	-		77		M	109		
14	N	n	46	.		78		N	110		
15	O	o	47	/		79		O	111		
16	P	p	48	0		80		P	112		
17	Q	q	49	1		81		Q	113		
18	R	r	50	2		82		R	114		
19	S	s	51	3		83		S	115		
20	T	t	52	4		84		T	116		
21	U	u	53	5		85		U	117		
22	V	v	54	6		86		V	118		
23	W	w	55	7		87		W	119		
24	X	x	56	8		88		X	120		
25	Y	y	57	9		89		Y	121		
26	Z	z	58	:		90		Z	122		
27	[		59	;		91			123		
28	£		60	<		92			124		
29	]		61	=		93			125		
30	↑		62	>		94			126		
31	←		63	?		95			127		

הקודים שבין 128 ל-255 הם הדמויות הנגטיביות של הקודים שבין 0 ל-127.

טבלה 2

הסימן	CHR$	הסימן	CHR$	הסימן	CHR$	הסימן	CHR$
	0	.	46	#	92	f4	138
	1	/	47	]	93	f6	139
	2	0	48	↑	94	f8	140
	3	1	49	←	95	SHIFT<RET'N>	141
	4	2	50		96	<UCASE>	142
<WHT>	5	3	51		97		143
	6	4	52		98	<BLK>	144
	7	5	53		99	<UCRSR>	145
<NOCBM>	8	6	54		100	<RVSOFF>	146
<OKCBM>	9	7	55		101	<CLR>	147
	10	8	56		102	<INST>	148
	11	9	57		103	<BRWN>	149
	12	:	58		104	<LTRED>	150
<RET'N>	13	;	59		105	<GRY1>	151
<LCASE>	14	<	60		106	<GRY2>	152
	15	=	61		107	<LTGRN>	153
	16	>	62		108	<LTBLU>	154
<DCRSR>	17	?	63		109	<GRY3>	155
<RVSON>	18	@	64		110	<PUR>	156
<HOME>	19	A	65		111	<LCRSR>	157
<DEL>	20	B	66		112	<YEL>	158
	21	C	67		113	<CYN>	159
	22	D	68		114	<SPACE>	160
	23	E	69		115		161
	24	F	70		116		162
	25	G	71		117		163
	26	H	72		118		164
	27	I	73		119		165
<RED>	28	J	74		120		166
<RCRSR>	29	K	75		121		167
<GRN>	30	L	76		122		168
<BLU>	31	M	77		123		169
<SPACE>	32	N	78		124		170
!	33	O	79		125		171
"	34	P	80		126		172
#	35	Q	81		127		173
$	36	R	82		128		174
%	37	S	83	<ORNG>	129		175
&	38	T	84		130		176
'	39	U	85		131		177
(	40	V	86		132		178
)	41	W	87	f1	133		179
*	42	X	88	f3	134		180
+	43	Y	89	f5	135		181
,	44	Z	90	f7	136		182
-	45	[	91	f2	137		183

הסימן	CHR$	הסימן	CHR$	הסימן	CHR$	הסימן	CHR$
▀	184	□	186	▝	188	▘	190
▄	185	▖	187	▘	189	▚	191

קודים 192-223 זהים ל-96-127

קודים 224-254 זהים ל-160-190

קוד 255 זהה ל-126

אינדקס

ספרי ארכימדס למחשבים

ארכימדס היא הוצאה לאור אשר שמה לעצמה למטרה להביא בפני קהל המשתמשים, הסטודנטים וחובבי המחשבים למיניהם, תרגומים של ספרים אשר זכו כבר להצלחה רבה בחו"ל.

בחירת הנושאים נעשתה באופן כזה, שהיא מהווה השלמה והרחבה לנושאים הנלמדים בסידרת "מחשבת".

לפניך רשימת הספרים שכבר יצאו לאור:

- שפת מכונה ואסמבלר ל-APPLE.
- שפת מכונה ואסמבלר ל-COMMODORE 64.
- שפת מכונה ואסמבלר ל-COMMODORE 128.
- שפת מכונה ואסמבלר ל-ATARI 800 XL, 130XE.
- מפת הזכרון ושגרות מערכת ההפעלה ל-COMMODORE 64.
- טורבו פסקל (מופיע בשני חלקים וכולל את גירסה 3.0).
- גרפיקה למתקדמים ל-COMMODORE 64.
- תכנות מתקדם: עיבוד נתונים ל-COMMODORE 128.
- שפת C.
- ספר כיס למשתמש: LOTUS 1-2-3.
- ספר כיס למשתמש DBASE III.
- DOS (MS ו-PC).

ובקרוב יצאו לאור:

- מדריך למעבד התמלילים Wordstar עברית-אנגלית
- שפת מכונה ואסמבלר ל-IBM-PC.
- המדריך השלם ל:LOTUS 1-2-3.
- ספר הפקודות השלם של שפת BASIC ל-IBM-PC ותואמיו.
- אנימציה מופלאה ל-IBM-PC.
- אנימציה מופלאה ל-APPLE.
- ספר כיס למשתמש: שפת מכונה למחשבי PC.

"ארנבת־64"
מערכת ממוחשבת ללימוד כתבנות

מבוא

חברתנו סיימה לפתח קורס ללימוד כתבנות בשפות עברית ואנגלית על הקומודור 64.

המערכת כוללת:
- תוכנה (על קסטה או על דיסקט)
- חוברת הדרכה ותמלילים.

התוכנה בנויה כך, שתוך כ-12 שעות לימוד עצמי תוכל כבר לתקתק על הקומודור שלך ועל כל מכונת-כתיבה או מעבד תמלילים תוך שימוש בכל 10 אצבעותיך.

אגב, שיטת הלימוד שפיתחנו נוסתה בהצלחה על מספר רב של לומדים.

שיטת הלימוד

חילקנו את הלימוד ל-20 שעורים. להערכתנו משך כל שעור הוא 30-15 דקות.

בכל שעור תלמד לתקתק 2 סימנים חדשים.

לכל שעור ישנם מספר אופציות ללימוד:

- **לימוד בסיסי:** תקתוק סימנים חדשים תוך שילובם בסימנים קודמים שלמדת.
- **מירוץ נגד ה"ארנבת":** זהו משחק קצר: "הארנבת" רצה לאורך השורה ועליך להשיג אותה - אתה מתקדם רק אם הקשת על אות נכונה.
- **מבחן־עצמי:** כאן עליך לתקתק בדייקנות 3 שורות. בסיום המחשב יודיע לך גם מה הקצב שבו כתבת את המבחן.
- **מעבד־תמלילים:** במצב זה עליך לתקתק על המסך את אחד הטקסטים המופיעים בחוברת. בגמר כתיבת הטקסט המחשב בודק אותו ומודיע לך אם נפלו בו שגיאות כדי שתוכל לתקן. אם אין שגיאות - המחשב מודיע לך על מהירות הכתיבה שלך.

קומודור 64

גרפיקה מתקדמת

ספר זה ילמד אותך כיצד לנצל את תכונות הגרפיקה המתקדמות של הקומודור 64. הוא יוצא מתוך הנחה שאתה בקיא למדי בתכנות בייסיק.

כל עקרון חדש מוצג צעד אחר צעד, החל ב-PEEK וב-POKE (פרק ראשון) וכלה בהתנגשות ספרייטים (פרק חמישי). תוך כדי כך תחקור מיפוי סיביות, מיפוי סיביות רב־צבעי מצב צבעים מורחב, שרטוט אותיות וספרייטים. בכל שלב מובהרות הנקודות על ידי דוגמאות ברורות.

ארבע תוכניות עיקריות מפותחות בספר: שני משחקים ושתי תוכניות שירות. המשחק הראשון נקרא "BREAKOUT", משחק של כדור ומחבט, שמשמש להכרת השימוש במיקום הצבע והמסך בזכרון הקומודור 64. המשחק השני נקרא "TARGET", והוא משתמש בספרייטים ליצירת משחק עתיר טכסיסים ואתגרים. שתי תוכניות השירות, "CHAR.GEN" ו-"SPRITE GEN" משמשות לתכנון תוים וספרייטים בהתאמה. מתכנתי קומודור 64, בעזרת מעט דמיון, יעריכו את העוצמה הכבירה שכלים אלה מציעים.

ארכימדס – הוצאה לאור